NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A196 812

DTIC FILE COPY

DTIC ELECTE AUG 1 5 1988 S D D

THESIS

DYNAMIC STALL ANALYSIS UTILIZING INTERACTIVE COMPUTER GRAPHICS

by

Eric L. Pagenkopf

March 1988

Thesis Advisor M.F. Platzer
Co-Advisor L.W. Carr

88

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report
2b Declassification Downgrading Schedule	Approved for public release; distribution is unlimited.
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School; 6b Office Symbol (if applicable) 67	7a Name of Monitoring Organization Naval Postgraduate School
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000	7b Address (city, state, and ZIP code) Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization; 8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number
8c Address (city, state, and ZIP code)	10 Source of Funding Numbers: Program Element No; Project No; Task No; Work Unit Accession No

11 Title (include security classification) DYNAMIC STALL ANALYSIS UTILIZING INTERACTIVE COMPUTER GRAPHICS

12 Personal Author(s) Eric L. Pagenkopf

13a Type of Report	13b Time Covered	14 Date of Report (year, month, day)	15 Page Count
Master's Thesis	From To	March 1988	90

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)
Field	Group	Subgroup	dynamic stall, computer graphics, IRIS, flow visualization, Theses.

19 Abstract (continue on reverse if necessary and identify by block number)

A Navier-Stokes problem solver, developed by L. N. Sankar, is modified to provide dynamic, interactive graphical presentations of predicted flow field solutions about a NACA-0012 airfoil section, oscillating in pitch, in order to demonstrate the capabilities of dynamic graphics applications in the study of complex, unsteady flows. Flow field solutions in the form of pressure coefficient and stream function contour plots about an airfoil experiencing dynamic stall are plotted utilizing an IRIS 3000-series workstation and Graphical Animation System (GAS) software, developed by Sterling Software for NASA. These full cycle solutions, in conjunction with dynamic surface pressure distribution plots and integrated lift, pitching moment and drag coefficient data, are compared to existing experimental data in order to provide an indication of the validity of the code's far-field solution. Full procedural documentation is maintained in order to provide an efficient analysis tool for use in future oscillating airfoil studies planned by the NASA-Ames Fluid Mechanics Laboratory and the Naval Postgraduate School Department of Aeronautics and Astronautics. Keywords

20 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified
22a Name of Responsible Individual M.F. Platzer	22b Telephone (include Area code) (408) 646-2311; 22c Office Symbol 54Ps

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. — security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

Dynamic Stall Analysis Utilizing
Interactive Computer Graphics

by

Eric L. Pagenkopf
Lieutenant, United States Navy
B.A., California State University, Northridge, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1988

Author: ______________________
Eric L. Pagenkopf

Approved by: M.F. Platzer
M.F. Platzer, Thesis Advisor

Lawrence W Carr
L.W. Carr, Co-Advisor

M.F. Platzer
M.F. Platzer, Chairman,
Department of Aeronautics and Astronautics

G E Schacher
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

A Navier-Stokes problem solver, developed by L. N. Sankar, is modified to provide dynamic, interactive graphical presentations of predicted flow field solutions about a NACA-0012 airfoil section, oscillating in pitch, in order to demonstrate the capabilities of dynamic graphics applications in the study of complex, unsteady flows. Flow field solutions in the form of pressure coefficient and stream function contour plots about an airfoil experiencing dynamic stall are plotted utilizing an IRIS 3000-series workstation and Graphical Animation System (GAS) software, developed by Sterling Software for NASA. These full cycle solutions, in conjunction with dynamic surface pressure distribution plots and integrated lift, pitching moment and drag coefficient data, are compared to existing experimental data in order to provide an indication of the validity of the code's far-field solution. Full procedural documentation is maintained in order to provide an efficient analysis tool for use in future oscillating airfoil studies planned by the NASA-Ames Fluid Mechanics Laboratory and the Naval Postgraduate School Department of Aeronautics and Astronautics.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The research for this thesis was conducted at the Fluid Mechanics Laboratory of the NASA Ames Research Center in conjunction with the Navy-NASA Joint Institute of Aeronautics and Astronautics. To Sanford Davis, Chief, FML, go my thanks for generously providing his facilities and personnel. I would also like to thank Max Platzer, Chairman of the NPS Department of Aeronautics and Astronautics, and Larry Carr and Satya Bodapati of NASA, for their guidance, advice and interest in this project. Sincere thanks for their 'nuts and bolts' advice go to NASA and Sterling Software employees, Joan Thompson, Ron Langhi, Rosalie Lefkowitz, Charlie Hooper, Greg Howe, Donna Diebert and especially Patty Askling, who willingly took on the time-consuming project of transferring my graphics to video tape. Finally, special thanks to my wife, Mia and son, Chris, who, for some reason, were more than happy to regularly send me shuffling off to the library or NASA during the past nine months.

I. INTRODUCTION

It has been recognized for some time [Ref. 1] that an airfoil oscillating in pitch to angles of attack greater than the static stall angle will surpass the traditional stall barrier and generate normal forces which exceed those attainable in the static case. This dynamic stall phenomenon is attributed to the aft movement of a strong shed vortex along the upper surface of the airfoil, carrying with it an induced velocity field which radically and dynamically changes chordwise pressure distributions. In general, an accurate interpretation of the dynamic stall mechanism will significantly impact a variety of applications, all of which involve dynamic lifting surface motions and unsteady flow separation. Originally, dynamic stall analysis efforts were directed toward helicopter aerodynamics, where sharp increases in oscillatory torsional loading and thus, blade stress, can reduce the fatigue life of rotor mechanical components and vortex-induced aerodynamic loading can generate adversely phased pitching moments, resulting in stall flutter [Ref. 2]. Much current interest concerns the feasibility of exploiting dynamic stall forces for effective, sustained maneuvering in the high angle of attack, "post-stall" flight regime, or supermaneuverability, for next-generation fighter and attack aircraft [Ref. 3]. Historically, attempts to analyze such complex, unsteady behavior relied heavily on empirical data, obtained from often laborious and time-consuming tests. With the advent of the supercomputer, however, computation of actual viscous flow fields about moderately complex computational models can now be numerically achieved in a matter of minutes through solution of the Reynolds-averaged Navier-Stokes equations [Ref. 4]. These solutions, in and of themselves, however, are not sufficient to promote insight into the mechanics and physics involved in such flows. This additional requirement, for effective visual portrayal of the flow field, is satisfied by application of high performance interactive computer graphics workstations and associated software.

The long range goal of the Computational Fluid Dynamics field is the development of a thoroughly verified computer code for unsteady aerodynamics which, among a myriad of other applications, will provide future aircraft designers with the opportunity to derive full advantage from application of the dynamic stall phenomenon. To this end, the Fluid Mechanics Laboratory (FML) of the NASA-Ames Research Center (ARC), in conjunction with the Naval Postgraduate School Department of Aeronautics and

Astronautics has planned an assortment of parallel and complementary oscillating airfoil windtunnel experiments and computer simulations.

The current study utilizes existing dynamic graphics packages to generate real-time projections of flow fields about a NACA-0012 airfoil oscillating in pitch and experiencing deep dynamic stall, as provided by a Navier-Stokes solver developed by L. N. Sankar [Refs. 5,6]. A modified version of the Sankar code was submitted via the FML front end VAX, to the NASA-ARC Cray X-MP 48 computer, which output flow field solutions at specified intervals throughout the oscillatory cycle. From this data, graphics files were generated which, in turn, were submitted to the Graphics Animation System (GAS) software as developed by Sterling Software under contract to NASA-ARC, and run on an IRIS 3000-series graphics workstation. (Final output, for demonstrative purposes, was then transferred to video tape.) Thus, the results of the study were two-fold. First, procedures by which interactive computer graphics could be efficiently (in terms of both computer-time and man-hours) and effectively (in terms of information display) incorporated as an analysis and verification tool for future FML studies, were developed, tested and refined. Secondly, the resultant graphics were utilized as a tool for the ongoing verification of the Sankar code.

II. DESCRIPTION OF THE SANKAR CODE

Simulation of complex phenomena occurring in real fluid flows requires accurate solutions to the full Navier-Stokes equations. The Sankar Navier-Stokes solver, developed for Blade-Vortex Interaction (BVI) studies, solves the two-dimensional, unsteady, compressible Euler and Navier-Stokes equations in strong conservation form, utilizing an alternating direction, implicit method as the time marching algorithm. A body-fitted C-grid, with clustering in the normal direction is utilized to discretize the flow field. Turbulent shear stresses are simulated with a two-layer algebraic eddy-viscosity model, with modifications as described later. The code may thus be utilized for solution of steady or unsteady, inviscid or viscous and laminar or turbulent flows.

A. GOVERNING EQUATIONS

In Cartesian coordinates, the 2-D, unsteady, compressible Navier-Stokes equations in strong conservative form may be written as

$$\delta_t\vec{q} + \delta_x\vec{E} + \delta_y\vec{F} = Re^{-1}(\delta_x\vec{R} + \delta_y\vec{S}) \tag{2.1}$$

where $\vec{q}$, $\vec{E}$, $\vec{F}$, $\vec{R}$ and $\vec{S}$ are four element vectors with entries corresponding, in order, to the equations of continuity, momentum (x- and y-) and energy. If non-dimensionalized such that all values of length, velocity (u and v), density (ρ) and total energy per unit volume (e) are normalized with respect to section chord length (c), free stream speed of sound (a_∞), free stream density (ρ_∞) and $\rho_\infty a_\infty$, respectively, these vectors may be written as

$$\vec{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \quad \vec{E} = \begin{bmatrix} \rho \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix} \quad \vec{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix} \quad \vec{R} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ R_4 \end{bmatrix} \quad \vec{S} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ S_4 \end{bmatrix} \tag{2.2}$$

The Reynolds number (Re), pressure (p), speed of sound (a) and stress terms (τ_{ij}) are defined as follows.

$$Re = \frac{\rho_\infty M_\infty a_\infty c}{\mu} \tag{2.3}$$

$$p = (\gamma - 1)[e - 0.5\rho(u^2 + v^2)] \tag{2.4}$$

$$a^2 = \gamma(\gamma - 1)[e/\rho - 0.5(u^2 + v^2)] \tag{2.5}$$

$$\tau_{xx} = (\lambda + 2\mu)u_x + \lambda v_y \tag{2.6}$$

$$\tau_{xy} = \mu(u_y + v_x)$$

$$\tau_{yy} = (\lambda + 2\mu)v_y + \lambda u_x$$

$$R_4 = u\tau_{xx} + v\tau_{xy} + kPr^{-1}(\gamma - 1)\delta_x a^2$$

$$S_4 = u\tau_{xy} + v\tau_{yy} + kPr^{-1}(\gamma - 1)\delta_y a^2$$

Assuming Stokes' hypothesis to be valid, the constant λ is defined as -2 3 μ . Since only airflow is considered, the ratio of specific heats (γ) is defined as 1.4 and the Prandtl number (Pr), as one.

Neglecting viscous terms results in the Euler equations, as the right-hand side of equation (2.1) goes to zero. Use of the strong conservative form (i.e., the continuity entry in $\bar{E}$, $\delta(\rho u)/\delta x$, is solved in its present form, vice the more mathematically correct form of $\rho\delta u/\delta x + u\delta\rho/\delta x$) allows the identical conservation of physical quantities (mass, momentum and energy) when finite difference schemes are applied.

B. TRANSFORMED GOVERNING EQUATIONS

The flow field region must be discretized into a transformed, finite difference mesh, or computational plane, in order to allow numerical solution of the governing equations. The most efficient grids are rectangular in shape with regular spacing or spacing gradients. They must, however, correspond to a flow field grid which provides high resolution (minimal spacing) in regions where gradients are large, particularly in the boundary layer and about the leading edge. In response to the coordinate transformation involved in the grid generation process, the governing equations, too, must be transformed. By defining ζ and η as functions of the cartesian coordinates (time is not transformed), this is simply a 2-D mapping procedure accomplished by application of the transformation Jacobian,

$$\frac{\delta(x,y)}{\delta(\zeta,\eta)} = \begin{bmatrix} \delta x/\delta\zeta & \delta x/\delta\eta \\ \delta y/\delta\zeta & \delta y/\delta\eta \end{bmatrix} \tag{2.7}$$

By setting

$$J = \frac{\delta(\zeta,\eta)}{\delta(x,y)} = \zeta_x\eta_y - \zeta_y\eta_x \tag{2.8}$$

$$J^{-1} = \frac{\delta(x,y)}{\delta(\zeta,\eta)} = x_\zeta y_\eta - x_\eta y_\zeta$$

the governing equation becomes

$$\delta_\tau q + \delta_\zeta E + \delta_\eta F = Re^{-1}(\delta_\zeta R + \delta_\eta S) \tag{2.9}$$

where

$$q = \vec{q}/J \tag{2.10}$$

$$E = (\zeta_t\vec{q} + \zeta_x\vec{E} + \zeta_y\vec{F})/J$$

$$F = (\eta_t\vec{q} + \eta_x\vec{E} + \eta_y\vec{F})/J$$

$$R = (\zeta_x\vec{R} + \zeta_y\vec{S})/J$$

$$S = (\eta_x\vec{R} + \eta_y\vec{S})/J$$

and

$$\tau_{xx} = (\lambda + 2\mu)(\zeta_x u_\zeta + \eta_x u_\eta) + \lambda(\zeta_y v_\zeta + \eta_y v_\eta) \tag{2.11}$$

$$\tau_{xy} = \mu[(\zeta_y u_\zeta + \eta_y u_\eta) + (\zeta_x v_\zeta + \eta_x v_\eta)]$$

$$\tau_{yy} = (\lambda + 2\mu)(\zeta_y v_\zeta + \eta_y v_\eta) + \lambda(\zeta_x u_\zeta + \eta_x u_\eta)$$

$$R_4 = u\tau_{xx} + v\tau_{xy} + kPr^{-1}(\gamma - 1)(\zeta_x\delta_\zeta a^2 + \eta_x\delta_\eta a^2)$$

$$S_4 = u\tau_{xy} + v\tau_{yy} + kPr^{-1}(\gamma - 1)(\zeta_y\delta_\zeta a^2 + \eta_y\delta_\eta a^2)$$

C. GRID GENERATION

Grid generation for discretization of the flow field region may be accomplished by conformal mapping, algebraic methods or numerically solving a set of partial differential equations. The original Sankar code utilizes either of the latter two methods, while the version currently vectorized for use in this study utilizes only the the algebraic method, which results in a body-fitted, sheared parabolic coordinate system or C-grid. Utilization of such body-fitted grids allows synchronous rotation of the entire grid and airfoil section for dynamic cases, using simple trigonometric relations. This coordinate system satisfies the general grid requirements for smoothness throughout and fine spacing in regions where high gradients exist, such as the boundary layer or leading edge.

Normalized geometric airfoil shape data, in Cartesian coordinates, is input in table format to the code, which utilizes an interpolative procedure to compute additional points and smooth the surface. The trailing edge region is modelled as a vortex sheet shape, or "cut", which smoothly leaves the airfoil, tangent to the mean camber line at the trailing edge. Algebraic manipulation of the section surface and cut allows grid generation in the transformed ξ - η plane. Figure 1 shows the resultant grid when mapped back to Cartesian coordinates.

Uniform spacing in the transformed plane's ξ -direction results in fine spacing at the leading edge in the real plane, but coarse spacing in the trailing edge region, due to uniform increments across the axis (as opposed to the standard C-grid which results in a region of finer spacing at the trailing edge). η -spacing in both planes increases, approximately exponentially, in directions normal to the airfoil surface, with the magnitude of initial spacing being user defined. Though easy to generate, this grid's effectiveness in the analysis of certain flow cases has proven somewhat limited, due to its coarseness in the trailing edge region [Ref. 6, p.44].

D. APPLICATION OF BOUNDARY CONDITIONS

Consideration of an airfoil impulsively started from rest in a fluid with uniform properties throughout, provides the initial conditions for solution of the parabolic Euler and Navier-Stokes equations (Eq. 2.9) in the computational plane. In addition, boundary conditions and artificial dissipation terms are required in order to achieve accurate solutions.

Three boundaries exist which are of interest, the surface, the far-field boundary (grid limit) and the trailing edge cut. Boundary conditions at the surface are driven by the no-slip condition which, in response to viscous effects, requires the fluid velocity at the

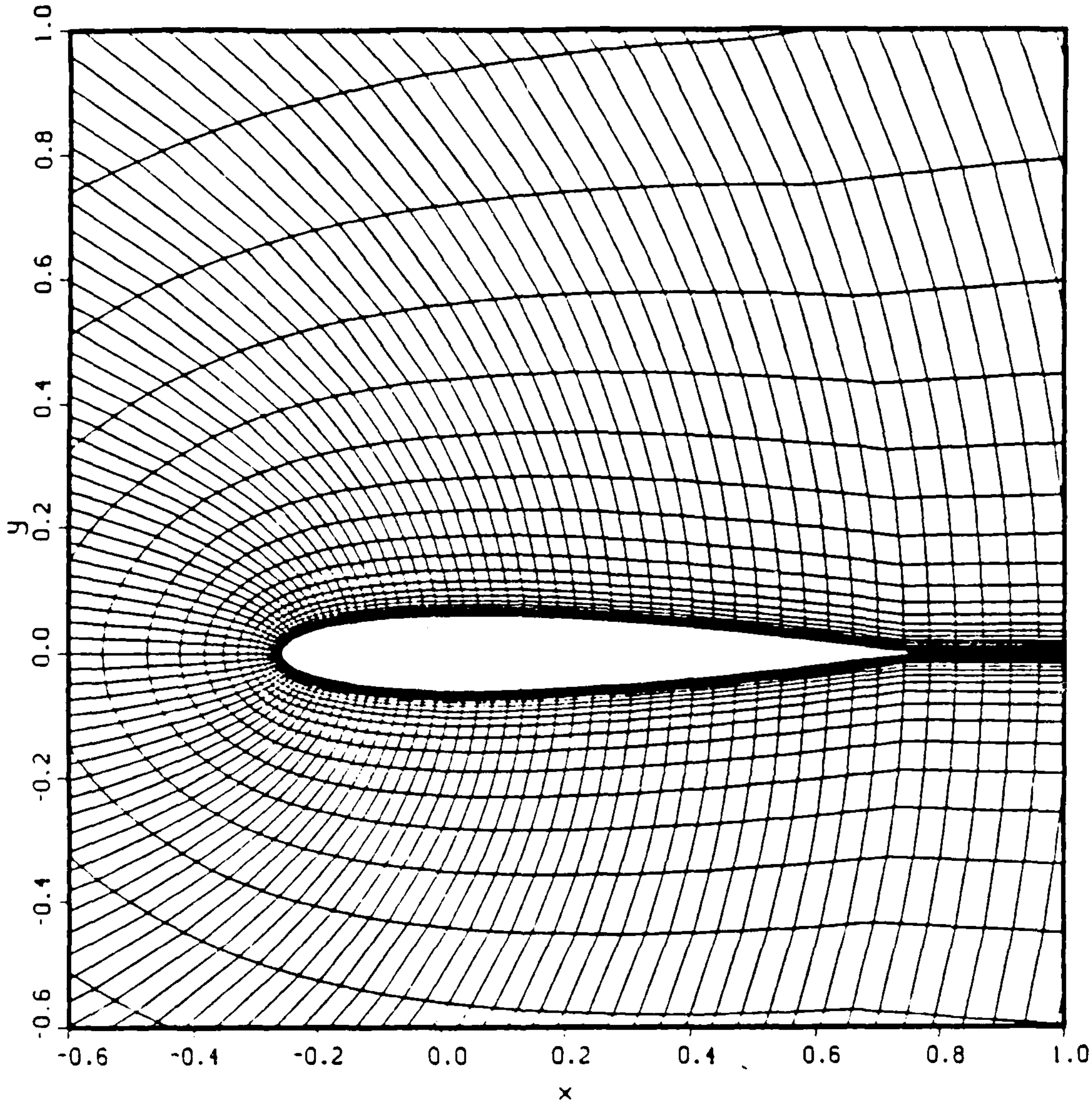

Figure 1. Algebraic C-Grid in Cartesian Coordinates

boundary to equal the boundary velocity. Thus, in this reference frame, u and v are set to zero on the solid surface. Since the surface is considered adiabatic, both $\delta e/\delta\eta$ and $\delta e/\delta\zeta$ are set to zero and, likewise, pressure distributions are determined from $\delta p/\delta\eta$ and $\delta p/\delta\zeta$ equal to zero at the solid boundary. (This is equivalent to neglecting stress contributions in the momentum equations, which is acceptable when dealing with high Reynolds numbers flows.) Boundary conditions at the cut are driven by the necessity to avoid discontinuities in the "continuous" portion of the flow field. Since the grid is extremely dense in the η -direction in this region, averages of the values of the two nearest interior points are assigned to points along the cut, allowing a smooth transition. Since the grid cannot economically be generated large enough to avoid disturbance velocities at the far-field boundary, boundary conditions must account for the presence of the airfoil. Linear small disturbance theory is applied to determine perturbation velocities at points along the boundary, which are then added to free stream conditions. Downstream boundaries are treated such that entropy changes can be convected out of the computational domain [Ref. 6, p. 29], allowing shocks and boundary-layer generated vorticity to pass through the grid.

It has been found that spatial derivatives are sensitive to the decoupling of odd and even points which necessarily occurs in central difference schemes. This results in the generation of high frequency errors in regions of large pressure gradients, such as are present in shocks or about stagnation points. Due to the high Reynolds numbers encountered in these flows, dissipation provided by the viscous terms are not enough to eliminate the errors, necessitating the addition of two artificial dissipation terms, embedded in the numerical schemes. An explicit artificial viscosity term is input by the user, with a proportional implicit term assigned by the code.

E. TURBULENCE MODELLING

Since during the derivation of the Navier-Stokes equations, no assumptions are made regarding flow type, these equations are instantaneously valid for both laminar and turbulent flows. The large range of time and spatial scales encountered in turbulent flows, however, makes solution of the instantaneous Navier-Stokes equations virtually impossible, at this time. As a result, the equations are time- or ensemble-averaged. Use of such equations, in response to turbulence, results in additional turbulent shear stress terms or the Reynolds stresses (named for Oswalde Reynolds, who initiated turbulence studies in the 1880's), which effectively are increases in shear stresses due to turbulent motion. The difficulties associated with the existence of additional unknowns without

any additional equations is described as the closure problem and is dealt with through the use of turbulence models. These are based on empirical knowledge of turbulent flows and thus, provided solutions will be approximate and require some confirmation from experiment [Ref. 7].

The most common turbulence modelling method involves mixing lengths, as developed by Prandtl. Assuming that turbulent fluctuations are essentially the result of velocity perturbations between adjacent streamlines, and that all fluctuating velocity components at a given point are of the same magnitude, it can be shown that turbulent or eddy viscosity (μ_T) is proportional to the magnitude of the local vorticity (ω) [Ref. 8, p. 387]. In Prandtl's mixing length model, the proportional term is the square of a characteristic length, related to fluid turbulence intensity. Prandtl suggested this characteristic length (l) be treated as

$$l = ky$$

where y is the normal distance from the fluid boundary and k is empirically obtained. Thus, the key to obtaining accurate solutions when utilizing models of this type lies in the mixing length expression.

The Baldwin-Lomax two-layer eddy viscosity model, based on Cebeci's two-layer model, is presently used in the Sankar code. This model divides the boundary layer into two regions, an inner layer and an outer layer, with separate methods for determining eddy viscosity used in each. The boundary for the two layers is defined as that point where eddy viscosities produced by the two methods match. The inner layer utilizes a mixing length model where eddy viscosity starts from zero at the wall and is defined by the following.

$$(\mu_T)_{inner} = \rho l^2 \,|\, \omega \,| \tag{2.12}$$

The mixing length term is defined by

$$l = \kappa y D \tag{2.13}$$

where y is the normal distance from the wall, κ is the Von Karman constant and D is the Van Driest damping function, given by

$$D = [1 - \exp(-y^+ / A^+)] \tag{2.14}$$

where A^+ is an empirical constant. The outer layer eddy viscosity model is defined by

$$(\mu_T)_{outer} = KC_{cp}\rho F_{wake}F_{Kleb}(y) \tag{2.15}$$

where K and C_{cp} are constants. The Klebanoff intermittency function, given by

$$F_{Kleb}(y) = [1 + 5.5(C_{Kleb}y/y_{max})^6]^{-1} \tag{2.16}$$

ensures that eddy viscosity approaches zero as the edge of the boundary layer is approached and the flow assumes external characteristics. C_{Kleb} is a constant. F_{wake} is a function defined the following relation.

$$F_{wake} = \min(y_{max}F_{max},\ C_{wk}y_{max}U_{dif}^2/F_{max}) \tag{2.17}$$

U_{dif} is the magnitude of the velocity profile's velocity range. F_{max} is the maximum value provided by the following.

$$F(y) = y\,|\,\omega\,|\,D \tag{2.18}$$

y_{max} is the value of y corresponding to F_{max}. Constants in the Sankar code are defined as follows:

$$A^+ = 26.0$$

$$\kappa = 0.4$$

$$K = 0.0168$$

$$C_{cp} = 1.6$$

$$C_{Kleb} = 0.3$$

$$C_{wk} = 0.25$$

In order to account for turbulence outside the boundary layer, such as that which occurs in the dynamic stall process, a modified turbulence model is available which effectively increases the outer model's mixing length. In this case, F_{max} and y_{max} are determined by the following.

$$F(y) = y^2\,|\,\omega\,|\,D \tag{2.19}$$

The modified turbulence model was developed in response to early predictions of flow separation at high angles of attack and has been found to cause premature reattachment during the downstroke of dynamic cycles. Its use, therefore, is only recommended until stall onset [Ref. 8,p. 33].

F. CODE STRUCTURE

The code is structured such that the following user defined parameters are input from logical unit 05 (appended to the end of the code for Cray processing): grid size, step size (dt), artificial viscosity magnitude, mean angle of attack (α_0), oscillation magnitude (α_1), angle for suspension of the modified turbulence model, reduced frequency (k), free stream Mach number (M_∞), Reynolds number (Re), distance of the first η - contour from the airfoil wall, starting time, pitch and restart flags and airfoil geometry data. Added to this list for the present study are the number of time steps for the code to march on the present run, the number of steps between each plot (output interval) and the total number of steps and plots completed on all previous runs. Variables are then initialized, previous stored solution datasets are retrieved and read (to allow incorporation of all datasets, including those from the present run, into a single, combined file) and flow field starting conditions are input. A series of subroutine calls then generate the grid (AIRFOL, SING, TABINT, WRAP), cluster grid points for viscous flows (CLUSTR, STRTCH), rotate the airfoil (ROTGRID) to its actual angle of attack and compute the initial metrics (METRIC).

At this point, the code is fully initialized for commencement of the flow field solution process, which is conducted via an iterative loop. At each iteration, time is first marched forward one time step, followed by computation of the time dependent values of angular velocity, angle of attack and step change of angle of attack, according to the relations:

$$\omega = 2kM_\infty \sin(2kM_\infty t) \tag{2.20}$$

$$\alpha_i = \alpha_0 - \alpha_1 \cos(2kM_\infty t)$$

$$\alpha_{i-1} = \alpha_0 - \alpha_1 \cos[2kM_\infty(t - dt)]$$

$$d\alpha = \alpha_i - \alpha_{i-1}$$

The grid is then rotated, in the physical plane, by applying the the following relations at each grid point.

$$x = x \cos(d\alpha) - y \sin(d\alpha) \tag{2.21}$$

$$y = y \cos(d\alpha) - x \sin(dsavepha)$$

Following recomputation of the metrics, the solution is computed by subroutine SLPS, the ADI-algorithm, which calls DISSIP, for computation of the explicit dissipation terms, STRESS and RESI, for computation of the inviscid and viscous terms, respectively, AMAT1 and MATRIX1, for computation of the Jacobian and its inverse in the ζ -direction and AMAT2 and MATRIX2, for computation of the Jacobian and its inverse in the η -direction. Subroutine STRESS also calls subroutine EDDY, the turbulence model, for computation of the viscosity coefficient. WALLBC enforces the wall boundary conditions and finally, solution files, as discribed later, are generated.

Prior to resumption of the loop, performance coefficients are generated by subroutines LOAD and CPPLOT. Surface pressure coefficients are obtained from the relation:

$$C_p = \frac{p_b - p_\infty}{1/2\rho_\infty(M_\infty a_\infty)^2} \tag{2.22}$$

where p_b is the surface pressure and p_∞ is the free stream pressure. Skin friction coefficients are a function of the wall shear stress (τ_w) according to the relations:

$$C_f = \frac{\tau_w}{1/2\rho_\infty(M_\infty a_\infty)^2} \tag{2.23}$$

$$\tau_w = Re^{-1} M_\infty W$$

$$W = u_y - v_x \cong J(\mu_\eta x_\zeta + v_\eta y_\zeta)$$

The aerodynamic loads of lift, drag and moment about the quarter-chord, are computed by the following relations.

$$C_l = C_n \cos(\alpha) - C_t \sin(\alpha) \tag{2.24}$$

$$C_d = C_n \sin(\alpha) - C_t \cos(\alpha)$$

$$C_m = \int C_p[(y - y_{c/4})dy + (x - x_{c/4})dx]$$

C_n and C_t are the normal and tangential forces obtained from summing surface pressure and skin friction forces according to the following.

$$C_n = \left[\int C_p dx + \int C_f dy \right] / c \tag{2.25}$$

$$C_t = \left[- \int C_p dy + \int C_f dx \right] / c$$

G. CODE MODIFICATIONS

The following modifications were made to the code, resident on the FML VAX, during the course of this study.

1. Solution output commands are placed within the loop in order to provide interval output.
2. Read commands are placed at the beginning of the main program in order to allow storage of all solutions (previous restarts and the current run) in a single combined file.
3. The airfoil section is rotated to the initial angle of attack, vice rotation of the free-stream direction.
4. Restarts may be made from Plot3D (output) format as well as vector format.
5. Additional Plot3D files (surface pressure and skin friction line plots) are output from subroutine CPPLOT.
6. Grid dimensions are made true variables.
7. User inputs are reformatted for ease of use.

III. DYNAMIC GRAPHICS GENERATION

A. INTERACTIVE COMPUTER GRAPHICS

Due to the complexities and time dependent phenomenon associated with unsteady flows, attainment of acceptable clarity in flow field solutions requires complete descriptive information across fine grids, at a large number of minute time intervals. Thus, analysis and visualization of data generated by codes such as Sankar's, is difficult due to both the volume of information provided and its temporal nature. Requirements identified for effective flow field visualization include maximization of the bandwidth of information transfer, such that it closely matches the capabilities of the human eye, maximization of the quality of graphical displays, by enhancing key features and suppressing others, and maximization of the controllability of information [Ref. 9]. Additional advantages in information transfer can be achieved through the use of redundant coding and structured displays [Ref. 10]. In response to these requirements, interactive computer graphics workstations have been evolved to complement the super-computer. Workstation capabilities, in terms of geometrical transformation and screen update dispatch have been utilized by programmers to produce effective representations of flow field motion, through synchronization of coordinated data sets. Solution clarity is enhanced by the high degree of spatial resolution and large range of colors afforded by the workstation displays. The interactive capabilities which currently exist for semi-complex flows serve to improve visual cues for display of three-dimensional data sets and allow immediate access to regions of interest.

B. HARDWARE

The NASA-ARC Fluid Mechanics Laboratory is currently equipped with an IRIS-3000 series workstation configured with 4Mbytes of display list memory and equipped with z-clipping and z-buffering hardware and a multiple key mouse. The IRIS display processing unit's refresh memory is arranged as a two-dimensional array (768 x 1024), with row-column positions matching display pixel x-y coordinates. 24 bits are reserved for each pixel at eight bits (256 intensities) per color, resulting in a range of 16,777,216 possible colors. When displaying dynamic graphics, the refresh memory is divided, in order to meet user demands, since flyback time (about 1.3 μ sec) is too short for complete picture update. This double buffering mode utilizes half the memory for refreshing the screen displays, while the other half is dynamically updated. This ensures

smooth motion on the display, but divides the number of bitplanes per pixel, in half. Thus, the number of colors available drops to 4096. In order to avoid a similar decrease in the range of colors available, color maps are utilized. In this case, pixel values in the refresh memory are routed to an index or color look-up table (with each entry composed of 24 predefined bits, which define the color), vice direct routing to the intensity digital-to-analog converter. The IRIS transformation rate, via a single, composed transformation matrix in homogeneous coordinates, is 80,000 coordinates per second, with points and lines generated at 3 million pixels per second, or 40,000 inches per second. Polygons can be filled (flat shading) at the rate of 44 million pixels per second, or 70,000 square inches per second. These capabilities allow generation of most displays at the rate of approximately 10 screens per second, which, under ordinary conditions, is greater than the fusion frequency of the human eye. Interactive transformations are ordinarily accomplished via the mouse.

C. PLOT3D SOFTWARE

Solutions provided by the Sankar code consist of multiple binary files, specifically formatted for input into the Plot3D graphics software, authored by Pieter Buning for NASA. Two filetypes exist, which are labeled as XYZ or grid files and Q or flow quantity files. Both types are three-dimensional matrices, with placement of data within the matrices corresponding to the row-column addresses of mesh points. XYZ-files have a depth of two in the two-dimensional case and provide cartesian coordinate definitions of each mesh point. Q-files have a depth of four in the two-dimensional case and provide computed flow quantities at each mesh point, corresponding to the q-vector of equation 2.1. Headers for each file list free-stream Mach, Reynolds number, angle of attack and time.

Additional flow field quantities at each mesh point are computed according to the following relations. [Ref. 11]

Definitions:

$$\gamma = 1.4$$

$$R = 1.0\ (Gas\ constant)$$

$$M = V/c$$

$$c = \sqrt{\gamma \frac{p}{\rho}}$$

Density:

$$\rho = \vec{q}(1) \qquad (3.1)$$

$$\rho_\infty = 1.0$$

$$\rho_0 = \rho\left[1 + \frac{\gamma - 1}{2} M^2\right]^{\frac{1}{\gamma - 1}}$$

Pressure:

$$p = (\gamma - 1)\rho[e_0 - .5V^2] \qquad (3.2)$$

$$p_\infty = \frac{1}{\gamma}$$

$$p_0 = \frac{p}{\gamma - 1}\left[1 + \frac{\gamma - 1}{2} M^2\right]^\gamma$$

Temperature:

$$T = \frac{p}{\rho R} \qquad (3.3)$$

$$\frac{T}{T_\infty} = \frac{p}{p_\infty} \frac{\rho}{\rho_\infty}$$

$$T_0 = T\left[1 + \frac{\gamma - 1}{2} M^2\right]$$

Enthalpy:

$$h = \gamma[e_0 - .5V^2] \qquad (3.4)$$

$$h_\infty = \gamma e_{0\infty}$$

$$h_0 = e_0 + \frac{p}{\rho}$$

Energy:

$$e_i = e_0 - .5V^2 \qquad (3.5)$$

$$e_0 = \frac{\vec{q}(4)}{\rho}$$

$$e_{\infty} = 1/(\gamma - 1)\frac{p_{\infty}}{\rho_{\infty}}$$

$$e_{0\infty} = e_{i\infty} + .5V_{\infty}^2$$

$$e_k = .5V^2$$

Entropy:

$$s = \ln\left[\frac{p}{p_{\infty}}\frac{\rho}{\rho_{\infty}}^{\gamma}\right] \tag{3.6}$$

$$s_0 = 0$$

Pressure Coefficient:

$$C_p = \frac{p - p_{\infty}}{.5\rho_{\infty}V_{\infty}^2} \tag{3.7}$$

$$C_{p0} = \frac{p_0 - p_{0\infty}}{.5\rho_{\infty}V_{\infty}^2}$$

Based on these values, the Plot3D software will produce plots of scalar functions (density, pressure, temperature, enthalpy, energy, velocity, entropy, momentum and shocks) suitable for contour plots, vector functions (velocity, vorticity, momentum and pressure gradient), particle trace functions (particle traces and vortex lines), shock wave locations and grid functions (computational meshes and walls).

Particle traces are generated using trilinear interpolation of values of the vector function inside a computational cell and second-order Runge-Kutta steps to advance the particle in space. Five steps are required for each cell. The algorithm for computing shocks computes the Mach number component in the direction of the local pressure gradient. Locations where this value decreases through 1.0 is plotted as a shock. Two-dimensional stream functions are calculated by integrating the mass flow across a coordinate line.

Output from Plot3D is available in a variety of formats and is a function of user-defined attributes. For dynamic plots, output is in the form of graphics files, formatted for further manipulation by animation software. Device independent plotting (DIP) is enabled through use of the ARCGraph binary file format which allows data to be manipulated by a collection of function libraries and utilities developed by the Advanced Computer Research Center at NASA-ARC. Parameter data, along with graphics primitives (device independent op-codes) are contained in these files, allowing data manipulation by the Cray, IRIS and attached VAXes. Plot3D generation of graphics files requires that several attributes be defined. In order to reduce user-tasking, Plot3D will initially search for a filename-specific initialization communications file, which contains all required inputs. Input files must be read singularly, which makes necessary the separation of those combined plotting files received from the Cray. Output graphics files, however, are once again stored in combined files in order to speed further processing. User-defined attributes include axis scales and extreme values, function, number of contours, line colors and types and wall definitions.

Subroutine CPPLOT, within the Sankar code, provides grid (XYZ) files which contain plotting information for surface pressure coefficient and skin friction coefficient line plots, corresponding to each output flow field solution file. These line plots are scaled and translated after separation from the combined files and plotted utilizing the grid function. Likewise, an angle of attack pointer plot, utilizing a sine wave format (generated external to the code), is also provided. Q-files containing dummy variables (not utilized for grid plot generation) are also provided to complement the line plot XYZ-files, as required by Plot3D.

D. GRAPHICS ANIMATION SYSTEM SOFTWARE

The final step in the visualization process is the animation or synchronization of coordinated graphics datasets. The Graphics Animation System (GAS) is a software package developed by Sterling Software under contract to NASA and provided to educational institutions throughout the country. It is device specific, running only on IRIS workstations, and is written in the C programming language under the UNIX operating system. Graphics files, in ARCGraph format, are read and stored in the IRIS' display list memory as objects by the menu-driven program and assembled according to sequence file instructions (stored transformation matrices). In order to smooth motion associated with geometric transformations, up to 30 linearly interpolated matrices are provided by the program, between user-identified keyframes. Titles, legends and object

attributes are also available. Ultimately, flow field representations may be transferred to video tape or 16mm film. Interactive viewing (ordinarily available in real-time) is controlled via a mouse, within the menu's "view data" window.

A complete description of GAS and associated video hardware may be found in Reference 12.

IV. RESULTS AND DISCUSSION

A. PROCEDURAL CONSIDERATIONS

Integration of the FML IRIS with related CFD software and the Cray X-MP 48 was tested by plotting the flow field about an airfoil experiencing deep dynamic stall, as provided by the Sankar code. The advantages of plotting dynamic stall are two-fold. First, it is an extremely complex and time-dependent phenomenon which is difficult to visualize from discrete data sets. Thus, the accuracy of solutions for this type flow field can only be determined by consideration of the complete flow field in both time and space. Dynamic graphics are therefore ideally suited for its study. Secondly, the FML is currently heavily involved in studies of dynamic stall, which include verification of the Sankar code. Cray output is thus used to full advantage.

In order to provide smooth animations, an extremely large number of plotting sets, provided at equal time intervals, must be generated by the code during the oscillatory cycle. Since access and transfer of this data is required several times during the graphics generation process, it was found that storage of datasets in combined files provided the greatest efficiency, especially in transfers between the Cray and IRIS. Software resident on the IRIS is then employed to separate the combined file into individual datasets. (Embedded in these programs are schemes to scale data, as necessary to control their ultimate on-screen positions, allowing, for example, placement of line plots in a specific corner of the display or plotting, for comparison, two flow fields side-by-side.) Individual datasets are automatically assigned names which correspond to those contained in Plot3D initiation files, also resident on the IRIS. These files currently contain commands for the processing of 50 datasets. Thus, the combined file separation and ARCgraph file production (Plot3X) process must be completed at intervals. The use of repetitive file names during this process, however, increases automation to such an extent that only a limited number of user inputs are required to complete the process. ARCgraph files as provided by Plot3D are again in combined file format, allowing easy (a single user command) input into animation software.

B. THE DYNAMIC STALL PROCESS

A prerequisite to review of any code-provided flow field solution is consideration of available empirical information. The following dynamic stall characteristics have been observed for both harmonically oscillating airfoils [Ref. 13] and airfoils undergoing

monotonic angle of attack increases [Refs. 14, 15]. First, the airfoil stalls at an angle of attack which exceeds the static stall angle. Second, the corresponding normal forces and pitching moments exceed those attainable in the static case and, finally, a rapid change in pitching moment magnitude, termed as "moment stall" by Harris and Pruyn [Ref. 16] occurs several degrees in azimuth prior to the normal force decrease, or "lift stall", vice simultaneous occurence as in the static case. The process which results in these characteristics can be broken down into a series of chronological events, as depicted in Figure 2 from Reference 13. While the specific characteristics for a given airfoil are a function of free-stream Mach number, Reynolds number, reduced frequency, mean angle of attack and oscillation amplitude, empirical data obtained from a NACA-0012 airfoil at k = .15, $Re = 2.5\times10^6$ and $\alpha = 15^\circ - 10^\circ \sin(\omega t)$ is provided in the following discussion to illustrate the dynamic stall process.

At point (a) of Figure 2, the static stall angle is exceeded without any detectable change in flow over the airfoil. The boundary layer remains thin with no evidence of flow reversal at the surface. At point (b), ($\alpha = 19 - 20^\circ$) flow reversal appears at the surface. Over the majority of the airfoil, the boundary layer remains thin and attached. At the rear portion of the airfoil, however, the boundary layer thickens as the flow gradually decreases to zero velocity. At point (c), large eddies appear in the boundary layer. By point (d), flow reversal has spread over much of the chord, from the trailing edge to $x/c = 0.3$. With as much as 50% of the airfoil experiencing flow reversal, no changes in C_n or C_m are detected. At point (e), ($\alpha = 23.4^\circ$) a vortex forms. The boundary layer at the front of the airfoil abruptly breaks down (simultaneously from $x/c = 0.0$ to 0.3) and the vortex moves downstream at approximately 35 - 45% of the free-stream velocity. At point (f) the lift-curve slope increases beyond the $2\pi\alpha$ limit of quasi-static flows. By point (g), the pressure distribution is altered sufficiently to produce a noticeable divergence in the pitching moment (moment stall) but is accompanied by a continued increase in lift. Maximum lift occurs at approximately mid-chord, followed by a sharp decrease (point (h)), which identifies lift stall. (Boundary layer separation has occurred to such an extent that continued increases in angle of attack will not result in continued increases in lift.) At point (i) ($\alpha = 24.95^\circ$), maximum negative moment occurs. The vortex moves off the trailing edge at $\alpha = 24.8^\circ$ (downstroke) and C_n and C_m move toward static levels. At point (j), the airfoil is fully stalled. At point (k), boundary layer flow begins to reattach, progressing from the leading edge at approximately 25 - 35% free-stream velocity and is complete by $\alpha = 7^\circ$ (downstroke). At point

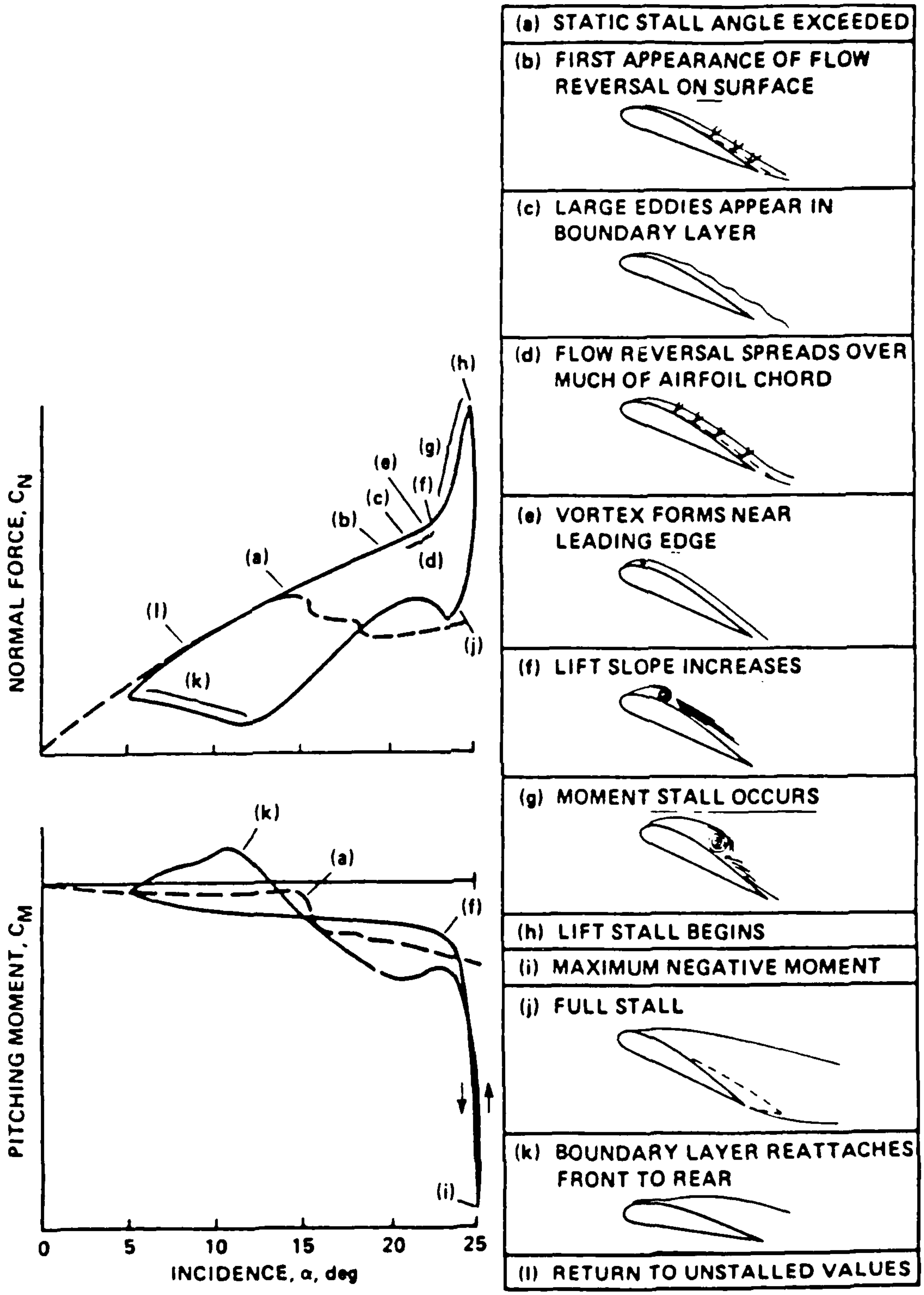

Figure 2. The Dynamic Stall Process (from Carr, Ref. 13)

(l), after some hysteresis ($\alpha = 6°$, upstroke) the separated region has fully closed and unstalled values of C_n and C_m are reestablished.

C. PROCEDURAL VERIFICATION

The case chosen for procedural verification closely corresponds to the Carr data of the previous section and exactly matches conditions previously run when the code was initially vectorized for use on the ARC Cray [Ref. 17]. There are several advantages associated with this duplication. First, it allowed storage of a complete record of this case (data saved at 300 intervals, vice 12) for future access. Second, it provided a simple means by which to determine if code modifications were correctly incorporated. Finally, it provided a baseline, in conjunction with empirical data, against which future sensitivity studies could be compared.

Inputs for this case were as follows:

$Re = 3.45 \times 10^6$

$M_\infty = .283$

$\delta t = .005$ sec.

η -min = .00005

ω = .151

Artificial Viscosity = 5

Grid size: 157 x 40

Oscillatory Cycle: $\alpha = 15° - 10° \cos(\omega t)$

The original Baldwin-Lomax turbulence model was used throughout the oscillatory cycle. Figures 3 through 7 are coefficient of pressure contour plots of the predicted flow field. (Dynamic plots are similar to these Plot3D-provided plots, but do not contain explicit quantitative information.) As previously noted [Ref. 17, p. 48], under these conditions, moment coefficients are consistently underpredicted as compared to experimental data. Drag and lift coefficients closely match experimental data below approximately 15 and 18 degrees angle of attack, respectively. Early prediction of flow separation, however, forces inaccuracies in quantitative solutions beyond these values and underprediction of the maximum lift coefficient obtainable. These findings highlight the rationale behind the modified turbulence model. Information provided by the dynamic plots, shows general qualitative agreement with experimental data, during the upstroke. A vortex forms near the leading edge of the airfoil section and progresses down its upper surface, gradually increasing in size. As the downstroke begins, however,

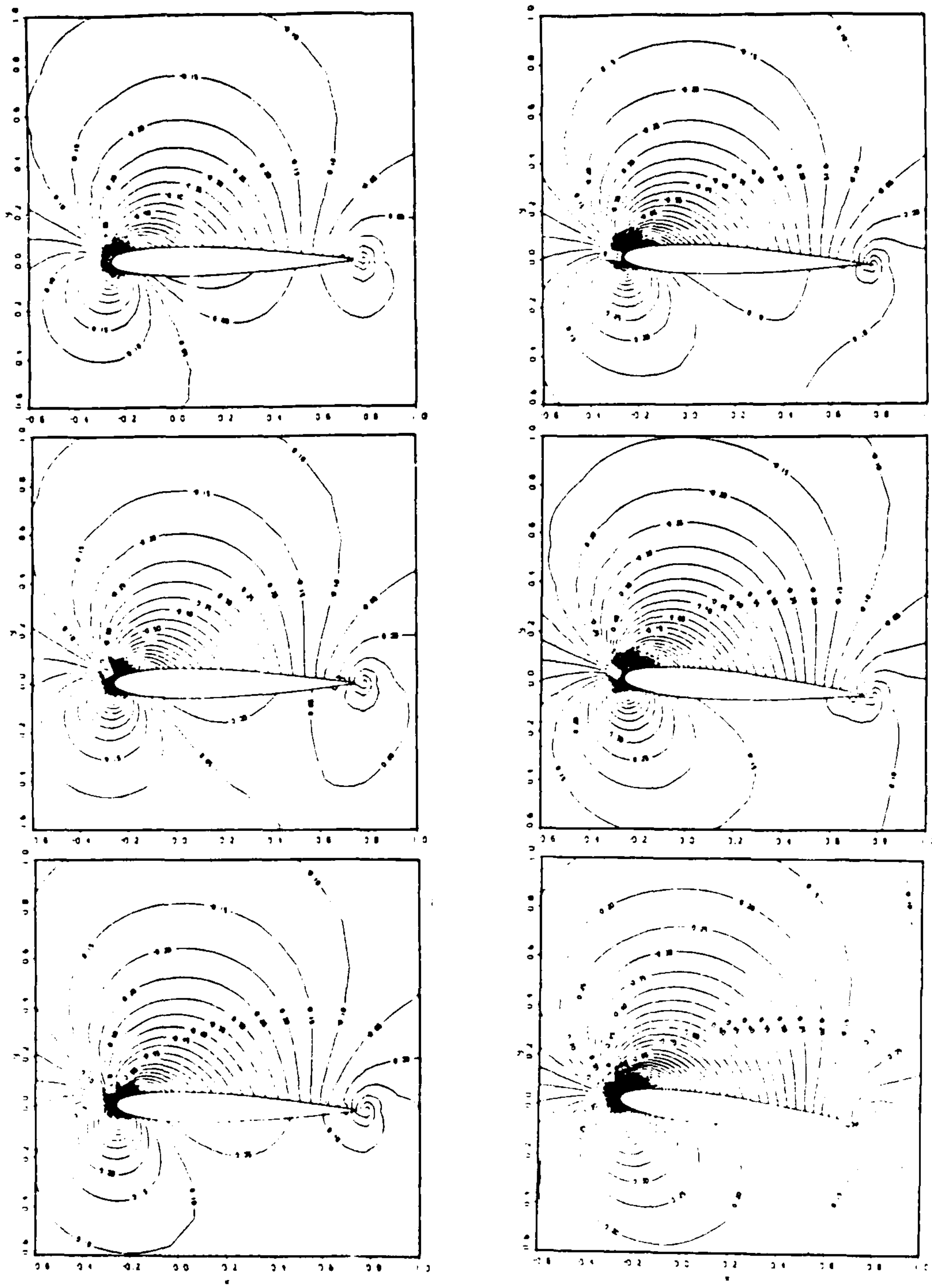

Figure 3. Pressure Contours, $\alpha = 15° - 10°$ (.151t), $\alpha = 5.00°, 5.24°, 5.92°, 7.02°, 8.47°, 10.23°$, (top to bottom, left to right), $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\eta_{min} = .00005$.

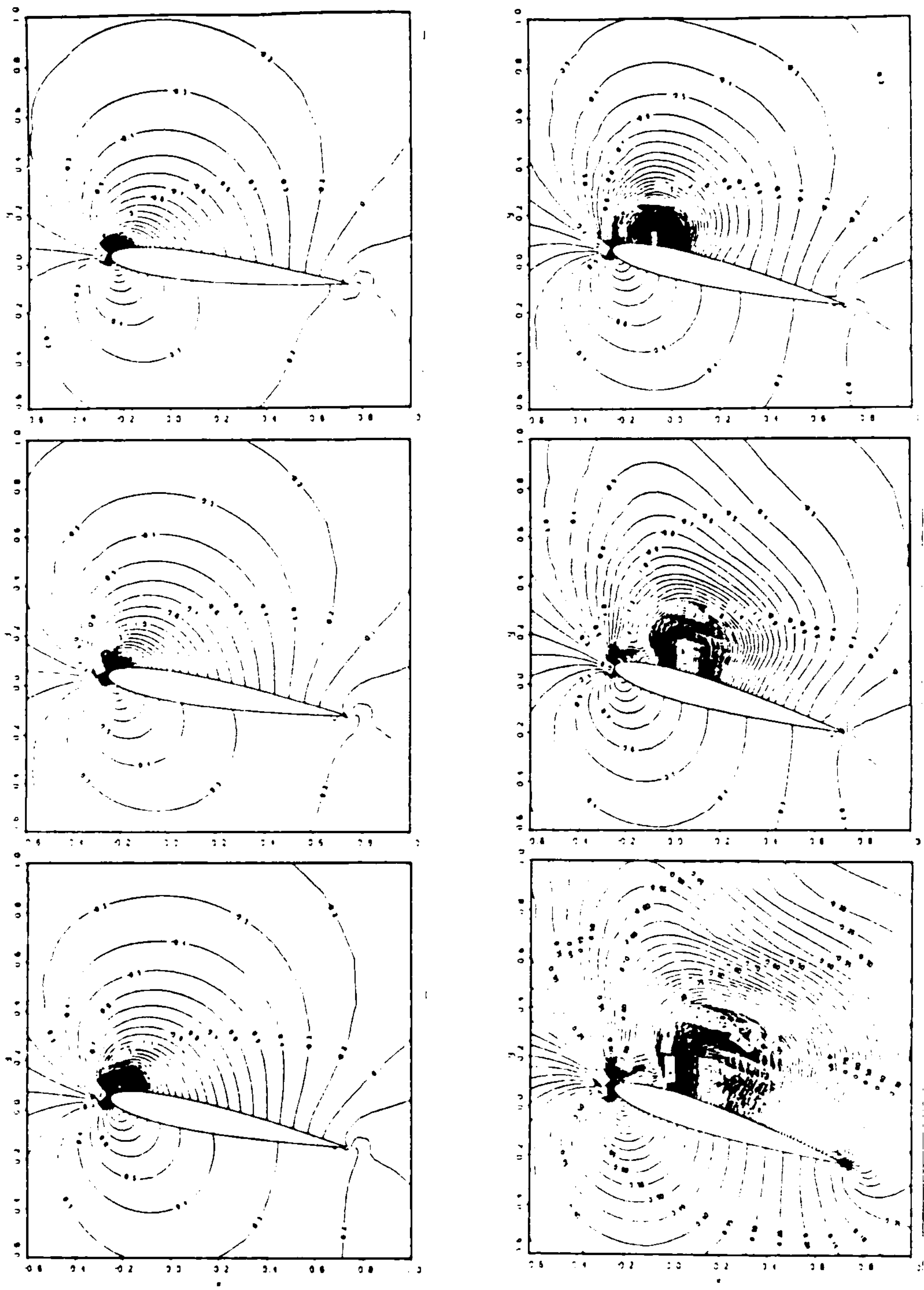

Figure 4. Pressure Contours, α = 15° - 10° (.151t), α = 12.20°, 14.30°, 16.43°, 18.50°, 20.40°, 22.07° , (top to bottom, left to right), $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\eta_{min} = .00005$.

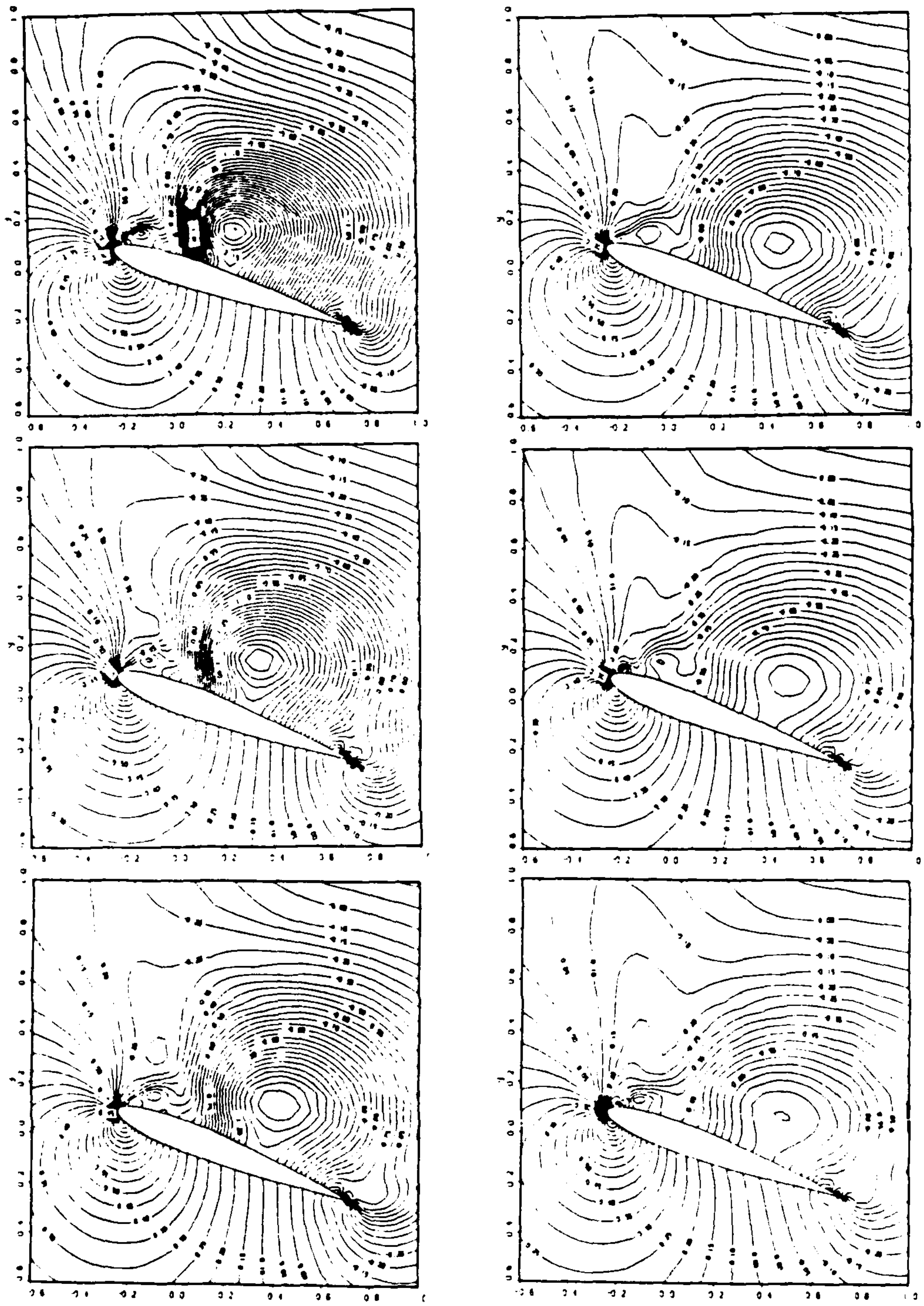

Figure 5. Pressure Contours, $\alpha = 15° - 10° (.151t)$, $\alpha = 23.41°, 24.36°, 24.89°, 24.98°, 24.60°, 23.80°$, (top to bottom, left to right), $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\eta_{min} = .00005$.

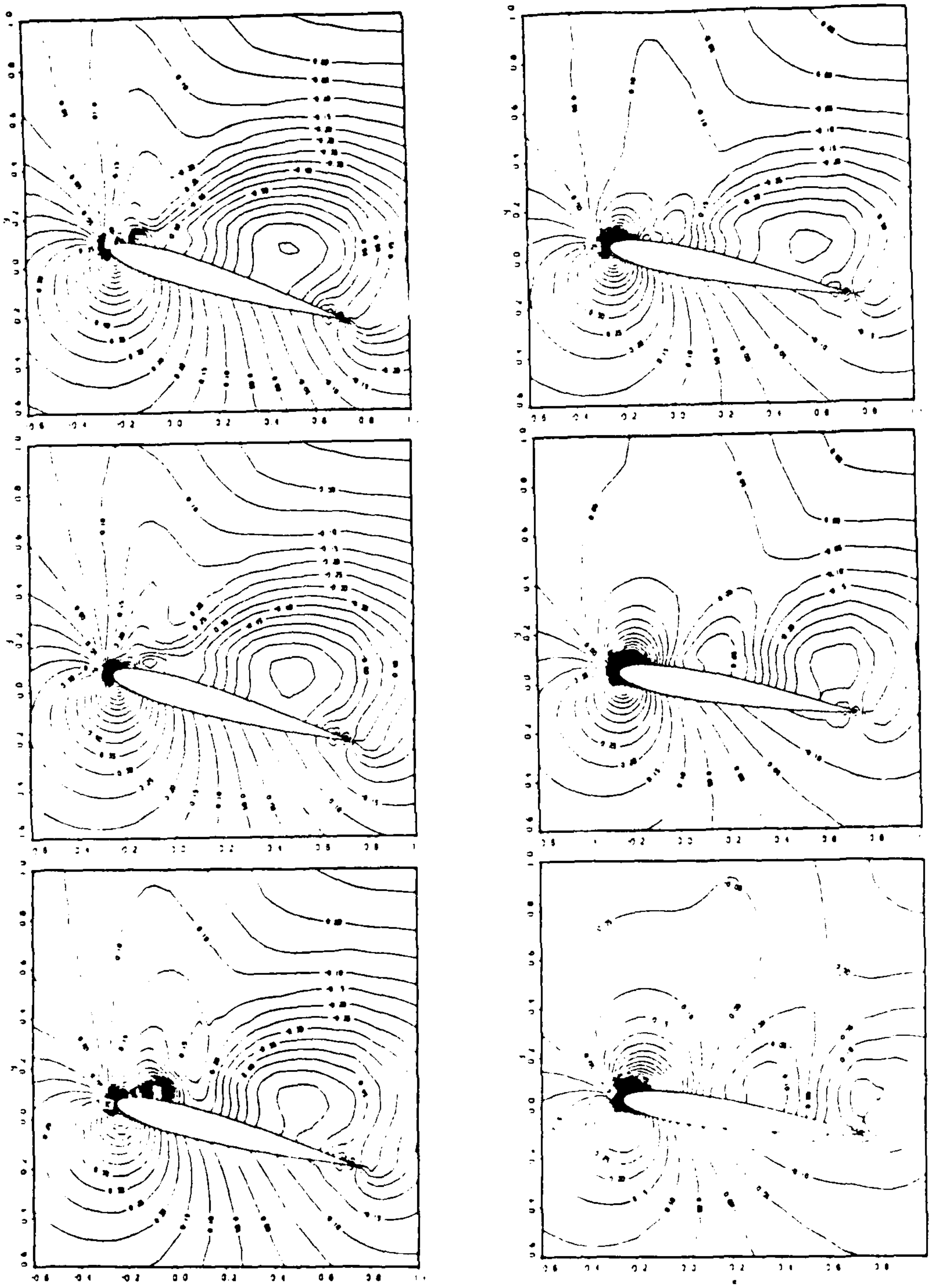

Figure 6. Pressure Contours, α = 15° - 10° (.151t), α = 22.59°, 21.03°, 19.20°, 17.19°, 15.07°, 12.94° , (top to bottom, left to right), $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\eta_{min} = .00005$.

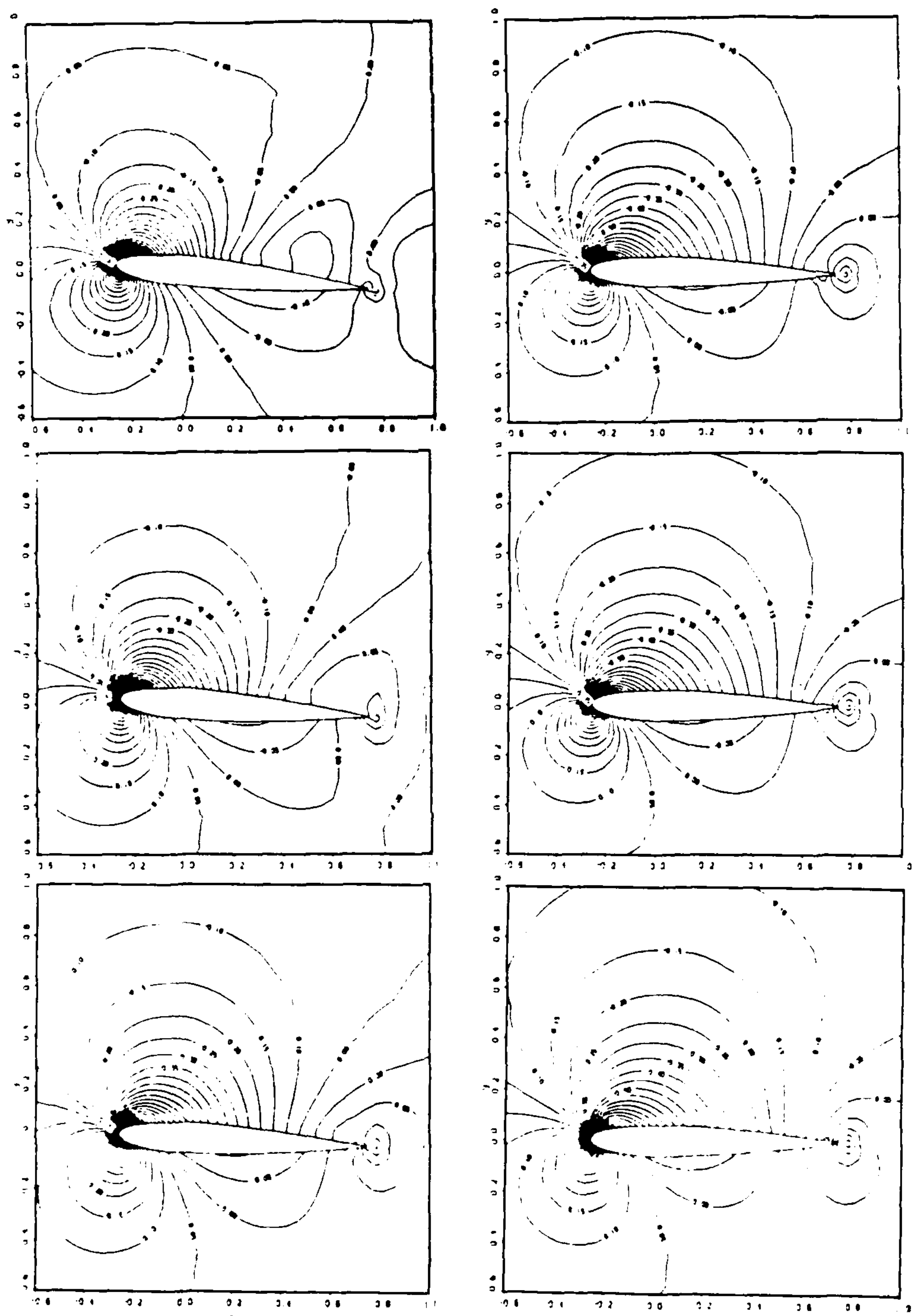

Figure 7. Pressure Contours, α = 15° - 10° (.151t), α = 10.92°, 9.07°, 7.50°, 6.27°, 5.43°, 5.00° , (top to bottom, left to right), $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\eta_{min} = .00005$.

rather than shedding from the trailing edge, the vortex stagnates at approximately the 65% chord point and gradually dissipates. Line plots of surface pressure coefficients more accurately portray (as compared to traditional carpet plots) the turbulent effects associated with the existence of the vortex. During the vortex dissipation phase, these line plots show a slight (gradually decreasing) pressure rise in the vicinity of the vortex. The subtle differences between these plots and those which would have been obtained had the vortex actually shed, are lost when this information is integrated for lift, drag and, to a lesser extent, moment coefficient data. Thus, quantitative data may not be good indicators of flow field solution accuracy.

D. SENSITIVITY ANALYSIS

The following code parameters which should effect dissipation of the vortex have been identified.

1. Artificial viscosity magnitude.
2. Turbulence model parameters.
3. Grid resolution.

A limited sensitivity analysis into the effects of grid resolution changes in the trailing edge region was conducted in order to provide direction for follow-on studies. During the initial phase of this analysis, only qualitative results will be considered, for those reasons mentioned above. Dynamic graphics, therefore, are ideally suited for this task, especially when data scaling capabilities are utilized in order to plot comparative datasets, side by side.

The grid is relatively coarse in the region in which the vortex becomes stationary and dissipates, as compared to those regions in which it behaves as expected. In the first sensitivity case, therefore, grid dimensions were held constant and the distance of the first constant- η line from the airfoil surface was increased to .001 , resulting in a finer grid in the trailing edge region. Figure 8 shows the effects of this change on the vortex. Within this grid, the vortex once again becomes stationary, but effectively splits, such that a secondary vortex is shed while the original vortex dissipates. While this does not correspond to any process encountered in actual flows, it does indicate that final solutions are sensitive to this parameter.

Changes in η -spacing have two effects on the flow field solution. Grid spacing in the trailing edge region becomes more fine at the expense of grid spacing in the boundary layer, which becomes more coarse. The extent of influence of the boundary layer sol-

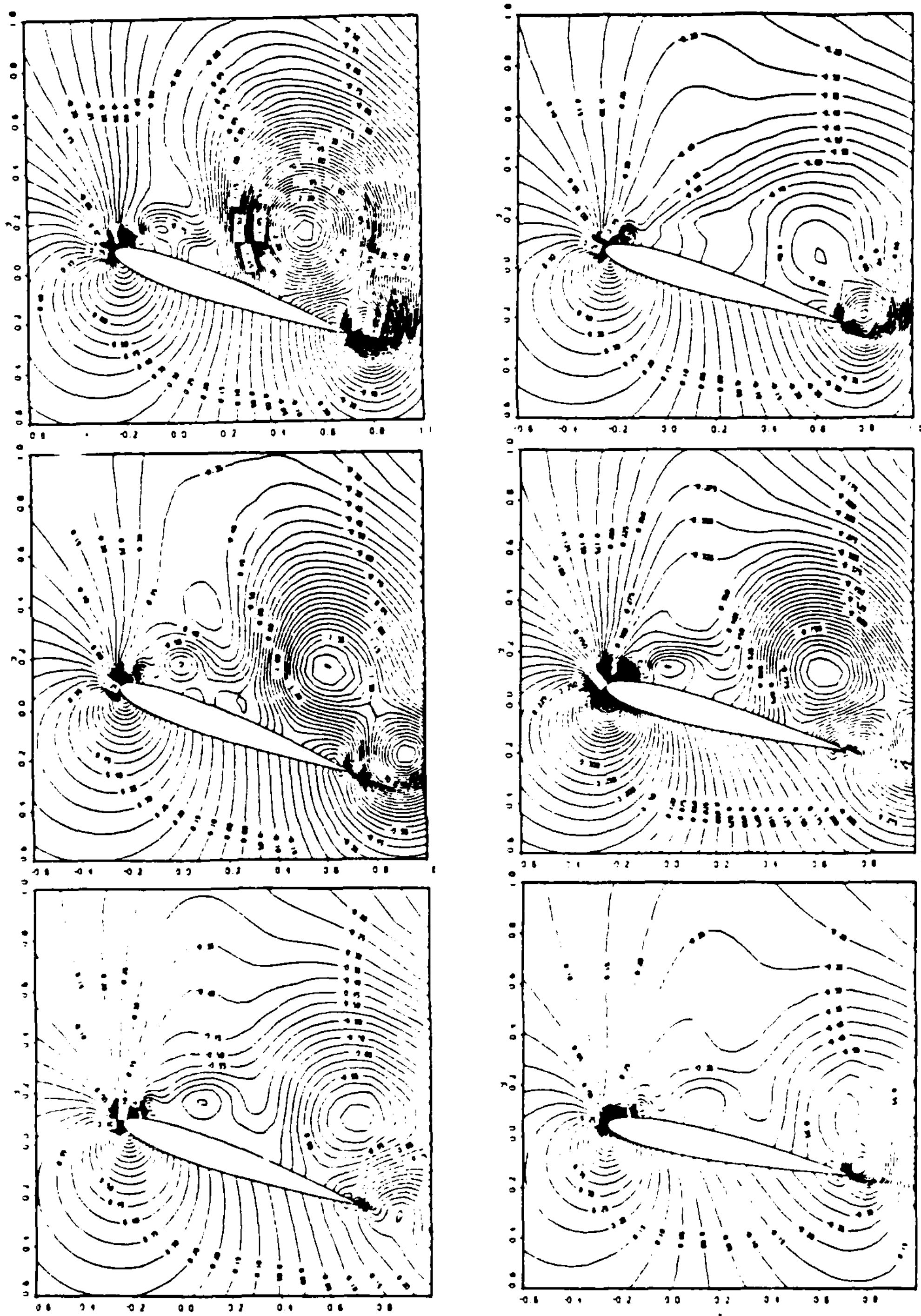

Figure 8. Pressure Contours, α = 15° - 10° (.151t), α = 24.89°, 24.85°, 23.80°, 21.85°, 19.20°, 16.13° , (top to bottom, left to right), $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\eta_{min} = .001$.

ution on the vortex, or far field solution, is not yet clear. It is known, however, that boundary layer solutions are extremely sensitive to η -spacing parameter changes, requiring an initial value of .00005 for accurate solutions [Ref. 18]. Further study, therefore, is required in order to determine whether vortex behavior changes observed in the previous case were in response to the altered boundary layer solution (which would recommend investigation of turbulence model parameters) or the altered grid. In order to facilitate this study, the existing grid would require enlargement (from 161 x 41). Unfortunately, this results in such a large memory allocation from the Cray, that it is prohibitively inefficient. Replacement of the entire grid generation process is therefore currently under consideration for this code.

E. CONCLUSIONS

The current study has resulted in completion of the following items.

1. Full procedural documentation (Appendix A) has been developed for efficient, code-independent, two-dimensional, dynamic graphics production on FML and NPS IRIS workstations. Data must be in Plot3D format and may be provided by either code or experiment. A method for simultaneous viewing of multiple datasets has been incorporated.
2. The Sankar Navier-Stokes solver (Appendix B) has been modified to allow dynamic graphical presentation of its flow field solutions.
3. Dynamic graphics applications in the study of complex flows were illustrated by means of an introductory sensitivity analysis, using the Sankar code. This limited analysis identified either grid resolution in the trailing edge region or the boundary layer solution as parameters to which vortex behavior is strongly dependent. (This suggests that incorporation of a more sophisticated grid generation scheme would be worthwhile.)

In order to derive full benefit from application of this technology, existing and future empirical data should be stored in plotting format, allowing the availability of visual records of multiple flow functions. Also, modification of additional Navier-Stokes solvers for dynamic output will allow effective code-to-code and code-to-experiment comparisons, all in an effort to eventually develop an accurate flow field predictor.

APPENDIX A. PROCEDURES FOR GENERATING DYNAMIC GRAPHICS AT FML

A. THE ANIMATION PROCESS

Creation of dynamic graphics for flow field visualization at the NASA-Ames Research Center Fluid Dynamics Laboratory involves five basic processes:

1. Navier-Stokes solver is submitted to the Cray X-MP 48 and dynamic output saved in the Central Storage Facility (CSF) in combined files.
2. Combined files are converted to IRIS binary and transferred to an FMLIRIS1 account.
3. Combined files are separated into individual plotting files on the IRIS and converted to combined graphics (.gra) files by Plot3D.
4. Graphics files are fed to the Graphics Animation System software (GAS) and dynamic graphics are plotted.
5. Resultant files and plots are transferred to storage or video tape.

Prior to beginning the animation process, the code must be modified to provide dynamic output in Plot3D format. This involves:

1. Placement of solution output commands within the iterative loop along with some interval flag.
2. If restarts are to be used, placement of read statements at the beginning of the program to allow storage of all provided solutions in a single combined file.
3. Storage of any additional required plotting information (line plots) in Plot3D XYZ files for eventual display utilizing Plot3D's grid function.

Full documentation of Plot3D formats and functions may be found in Sterling Software technical note 15, *A Guide to Generating Movies using Plot3D and GAS*, by Greg Howe. The procedures outlined below refer specifically to a version of the Sankar Navier-Stokes solver, modified for dynamic output. They can easily, however, be generalized to apply to any code (or experiment) from which dynamic output is desired. While many methods of completing these steps may exist, those outlined below have proven to be the most time efficient.

1. Vax Submitted Cray Jobs

A full dynamic cycle, utilizing a 161 x 41 grid, requires over 5000 seconds of Cray time, making job chaining necessary to obtain a reasonable priority. With 900 second jobs, 24 to 48 hours are usually sufficient for the full computation to be com-

pleted. Since significant differences exist between the first JCL from which Plot3D files are saved and subsesquent restart JCL's, two JCL's are maintained on identical copies of Sankar's code.

INITIAL.JCL. Accesses CSF or Cray tape (via STAGEX commands) for initial solution, which may be in either Plot3D or matrix format. Combined Plot3D files are initialized on CSF and assigned PDN's and ID's. Tape 05 input data (appended to end of code) is assigned in accordance with definitions included in MAIN portion of the code. (If restart is from dynamic data, AOA will not be a whole number. TSTART must be adjusted accordingly, in order to have accurate time AOA matches.)

RESTART.JCL. Tape 05 inputs must identically match those from INITIAL.JCL, with the exception of TSTART,CSTP,CPLT and possibly FORMAT. INITIAL.JCL-defined PDN ID solutions are accessed (read, stored and appended with additional solutions), saved (with identical PDN ID's) and scrubbed (oldest versions deleted). The initial restart solution is accessed with library routine GETP3D, written by Greg Howe, which will read a combined file, skip a specified number of datasets (DSSKIP) and access the next dataset (or datasets, if NUMDS is not equal to one, the default). Thus, if

DSSKIP + CPLT - 1,

the correct solution will be provided for restarts. Subsequent restarts illustrate the advantage of combined file name (PDN ID) repetition, as they simply require the following RESTART.JCL and Tape 05 updates:

1. DSSKIP
2. Job chain commands.
3. CSTP
4. CPLT

Once the optimum number of plotting sets for a cycle is determined, plotting files for an AOA pointer on IRIS displays can be generated using Plot3D. (500 sets maximum, or modify dimension statements.)

Line Plots. Any pointer or line plot information may be generated, either within the code or externally, and stored in XYZ files for further processing. Utilization of Plot3D function 0 and proper descriptors of walls will produce the desired plots. SINE.TXT is a program which generates plots for a dynamic AOA pointer in sine wave format. In this program, Tape 05 includes values for pointer scaling (axis length with

no scal· ·s 2π) and positioning on the IRIS screen and initial pointer location. For examp... the cycle starts from the minimum AOA.

PNLOCI = .75(NPLOTS).

Hardcopy printouts may also be obtained from these grid files by use of programs such as CPPLOT.TXT. This program provides printouts of upper and lower surface pressure coefficient and skin friction values at x c locations, including Cp plot generation.

2. Path to the IRIS

SEND.JCL. Utilizes GETP3D and SENDWKS to convert to IRIS binary and transfer complete or partial combined files, or individual plotting sets, from the CRAY to the IRIS. (Since I O's between the Cray and IR... can get "clogged" if too many transfers are requested, the individual method is not recommended.) This may need to be completed in stages to avoid exceeding the IRIS' me...ory capacity.

3. Creating Graphics Files

Combined files on the IRIS are separated into individual plotting sets by the programs **GETSIN.F** (also generates the dummy Q file, "qsin.iris"), **GETCP.F** (also scales and positions Cp or skin friction on the IRIS screen) and **GETX.F** (for XYZ and Q flow field datasets). These programs require some initial user inputs prior to running. Output dataset names will automatically match Plot3D initialization file names. Since the Plot3x run for each dataset type (X, P and SIN) requires specific arguments, initialization files for each data type exist separately in files XINI.COM, PINI.COM and SINI.COM. Since Plot3X searches for the file name 'PLOT3DINI.COM', these files must be renamed appropriately, utilizing the "mv" ... mand. Entering the command "plot3x" will then generate multiple graphics files which are stored in a combined file matching the initial Q file name encountered by Plot3D, (i.e., "q001.gra"). This file is then stored on an appropriate subdirectory or fed to GAS.

4. Notes on GAS

For movies, the maximum number of objects per sequence is fifty. The "seq(n).seq" files are 50 object far-field solution files (seq(n).seq = objects n(1-500), which use object 400 for titles. They can be fed to GAS (after object input) via the AUX I O window. For interactive viewing, the VIEW DATA window is the only usable option. (No sequence files are required in view data.) Full GAS documentation is available in the GAS User's Guide, available from Sterling Software.

5. Storage

Since IRIS memory space is severely limited at FML, resultant graphics files should be stored elsewhere, if not in immediate use. Downloading to 1 4" cassette tape is easily accomplished (also a recommended backup), as is transfer to CSF.

B. SUPPORT CODES

1. INITIAL.JCL

```
JOB,JN=FLYNAVY,T=900.  ERIC PAGENKOPF,x4269.
ACCOUNT,AC=,US=,UPW=.
*.
CFT,ON=A,OFF=S.
*.
*.RESTART COMMANDS:------------------------------------------------------
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=OLDSLN,PDN=NSE509,ID=VALDES.
ASSIGN,DN=OLDSLN,A=FT07.
*.-----------------------------------------------------------------------
*.SAVE DATA FROM PRESENT RUN IN COMBINED FILE:---------------------------
ASSIGN,DN=XYZ,    A=FT20.
ASSIGN,DN=Q,      A=FT21.
ASSIGN,DN=CPX,    A=FT50.
ASSIGN,DN=CFX,    A=FT60.
*.
LDR,MAP=PART,SET=ZERO.
*.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=XYZ,    PDN=SNKR05X,    ID=FLYNAVY.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=Q,      PDN=SNKR05Q,    ID=FLYNAVY.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=CPX,    PDN=SNKR05P,    ID=FLYNAVY.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=CFX,    PDN=SNKR05F,    ID=FLYNAVY.
*.-----------------------------------------------------------------------
*.ACCESS,DN=SENDVAX,PDN=SENDVAX,ID=STTRDM,OWN=RFTRDM.
*.SENDVAX,DN=XYZ,VDN='RAL"JIANPS password"::[JIANPS.PAGAN.DATA]XT1.DAT'.
*.SENDVAX,DN=Q,VDN='RAL"JIANPS password"::[JIANPS.PAGAN.DATA]QT1.DAT'.
*.ACCESS,DN=SENDWKS,PDN=SENDWKS,ID=STTRDM,OWN=RFTRDM.
*.SENDWKS,DN=Q3X49,VDN='FMLIRIS1"jiaa password"::"/u/jiaa/pagan/q.iris"',NOREC.
*.SENDWKS,DN=CFX,VDN='FMLIRIS1"jiaa password"::"/u/jiaa/pagan/cf01.dat"',NOREC.
*.-----------------------------------------------------------------------
*.JOB CHAINING COMMANDS:-------------------------------------------------
FETCH,DN=JOB2,TEXT='[PAGAN]NSMULT.TXT;2'.
REWIND,DN=JOB2.
SUBMIT,DN=JOB2.
*.-----------------------------------------------------------------------
*.
/EOF
```

2. RESTART.JCL

```
JOB.JN=FLYNAVY,T=900.  ERIC PAGENKOPF,X4269.
ACCOUNT,AC=,US=,UPW=.
*.
CFT,ON=A,OFF=S.
*.
*.RESTART COMMANDS:--------------------------------------------------
*.CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=OLDSLN,PDN=NSE509,ID=VALDES.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=GETP3D,  PDN=GETP3D,  ID=RFRICC.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=XYZ1,    PDN=SNKR05X,ID=FLYNAVY.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=Q1,      PDN=SNKR05Q,ID=FLYNAVY.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=CPX1,    PDN=SNKR05P,ID=FLYNAVY.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=CFX1,    PDN=SNKR05F,ID=FLYNAVY.
REWIND,DN=Q1.
GETP3D,I=Q1,O=OLDSLN,DATATYPE=Q,DSSKIP=49.
REWIND,DN=OLDSLN.
REWIND,DN=XYZ1.
REWIND,DN=Q1.
REWIND,DN=CPX1.
REWIND,DN=CFX1.
ASSIGN,DN=XYZ1,   A=FT90.
ASSIGN,DN=Q1,     A=FT91.
ASSIGN,DN=CPX1,   A=FT92.
ASSIGN,DN=CFX1,   A=FT93.
ASSIGN,DN=OLDSLN,A=FT07.
*.------------------------------------------------------------------
*.SAVE DATA FROM PRESENT RUN IN COMBINED FILE:-----------------------
ASSIGN,DN=XYZ,    A=FT20.
ASSIGN,DN=Q,      A=FT21.
ASSIGN,DN=CPX,    A=FT50.
ASSIGN,DN=CFX,    A=FT60.
*.
LDR,MAP=PART,SET=ZERO.
*.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=XYZ,    PDN=SNKR05X,    ID=FLYNAVY.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=Q,      PDN=SNKR05Q,    ID=FLYNAVY.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=CPX,    PDN=SNKR05P,    ID=FLYNAVY.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=CFX,    PDN=SNKR05F,    ID=FLYNAVY.
CSF,SCRUB,CSFACCT=,CSFPSWD=,PDN=SNKR05F,ID=FLYNAVY.
CSF,SCRUB,CSFACCT=,CSFPSWD=,PDN=SNKR05P,ID=FLYNAVY.
CSF,SCRUB,CSFACCT=,CSFPSWD=,PDN=SNKR05X,ID=FLYNAVY.
CSF,SCRUB,CSFACCT=,CSFPSWD=,PDN=SNKR05Q,ID=FLYNAVY.
*.------------------------------------------------------------------
*.ACCESS,DN=SENDVAX,PDN=SENDVAX,ID=,OWN=.
*.SENDVAX,DN=XYZ,VDN='RAL"JIANPS password"::[JIANPS.PAGAN.DATA]XT1.DAT'.
*.SENDVAX,DN=Q,VDN='RAL"JIANPS password"::[JIANPS.PAGAN.DATA]QT1.DAT'.
*.ACCESS,DN=SENDWKS,PDN=SENDWKS,ID=,OWN=.
*.SENDWKS,DN=Q3X49,VDN='FMLIRIS1"jiaa password"::"/u/jiaa/pagan/q.iris"',NOREC.
*.SENDWKS,DN=CFX,VDN='FMLIRIS1"jiaa password"::"/u/jiaa/pagan/cf01.dat"',NOREC.
*.------------------------------------------------------------------
*.JOB CHAINING COMMANDS:--------------------------------------------
FETCH,DN=JOB3,TEXT='[PAGAN]JOB3.TXT'.
REWIND,DN=JOB3.
SUBMIT,DN=JOB3.
*.------------------------------------------------------------------
*.
/EOF
```

3. SEND.JCL

```
JOB,JN=GETX,T=30. ERIC PAGENKOPF, X4269.
ACCOUNT,AC=,US=,UPW=.
*.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=XYZ,PDN=SNKR38X,ID=FLYNAVY.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=Q,PDN=SNKR38Q,ID=FLYNAVY.
*.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=GETP3D,PDN=GETP3D,ID=RFRICC.
ACCESS,DN=SENDWKS,PDN=SENDWKS,ID=,OWN=.
*.
REWIND,DN=XYZ.
REWIND,DN=Q.
*.
GETP3D,I=XYZ,O=XOUT1,DATATYPE=XYZ,DSSKIP=100,NUMDS=150.
GETP3D,I=Q,O=QOUT1,DATATYPE=Q,DSSKIP=100,NUMDS=150.
*.
SENDWKS,DN=XOUT1,VDN='FMLIRIS1"jiaa password"::"/u/jiaa/pagan/38x.iris"',NOREC.
SENDWKS,DN=QOUT1,VDN='FMLIRIS1"jiaa password"::"/u/jiaa/pagan/38q.iris"',NOREC.
*.
*.FETCH,DN=JOB2,TEXT='[PAGAN]SEND.JCL;2'.
*.REWIND,DN=JOB2.
*.SUBMIT,DN=JOB2.
*.
```

4. SINE.TXT

```
JOB,JN=FLYNAVY,T=30.  ERIC PAGENKOPF,X4269.
ACCOUNT,AC=,US=,UPW=.
*.
ACCESS,DN=SENDWKS,PDN=SENDWKS,ID=,OWN=.
*.
CFT,ON=A,OFF=S.
*.
ASSIGN,DN=X,  A=FT50.
ASSIGN,DN=Q,  A=FT60.
*.
LDR,MAP=PART,SET=ZERO.
*.
SENDWKS,DN=Q,VDN='FMLIRIS1"jiao PASSWORD"::/u/jiao/pagan/qsin.iris',NOREC.
SENDWKS,DN=X,VDN='FMLIRIS1"jiao PASSWORD"::/u/jiao/pagan/snkr05s',NOREC.
CSF,SAVE,CSFACCT=,CSFPSWD=,DN=X,PDN=SNKR05S,ID=FLYNAVY.
*.
/EOF
      PROGRAM POINTER
***********************************************************************
*                                                                     *
*     THIS PROGRAM GENERATES PLOT3D FILES WHICH WILL PRODUCE AN       *
*     AOA POINTER WHEN FUNCTION 0 IS SELECTED. XYZ FILES ARE SAVED    *
*     ON CSF FOR FUTURE ACCESS VIA PROGRAM GETSIN.JCL. A SINGLE Q     *
*     FILE OF PROPER DIMENSION IS SENT DIRECTLY TO THE IRIS WITH      *
*     FILENAME QSIN.IRIS.                                             *
*                                                                     *
*     VARIABLES:                                                      *
*          NPLOTS: TOTAL NUMBER OF PLOTS IN ONE CYCLE.                *
*          PNLOCI: INITIAL INTEGER LOCATION OF POINTER                *
*          PNLOC: INTEGER LOCATION OF POINTER.                        *
*          XSCALE: AXIS SCALING FACTOR                                *
*          YSCALE: SINE WAVE SCALING FACTOR                           *
*          XPOSIT: X-POSITIONAL FACTOR (SCREEN LOCATION)              *
*          YPOSIT: Y-POSITIONAL FACTOR (SCREEN LOCATION)              *
*                                                                     *
***********************************************************************
C
      INTEGER PNLOC,GDIM,PNLOCI
      DIMENSION SINX(3,500),SINY(3,500),Q1(3,500),Q2(3,500)
      DIMENSION Q3(3,500),Q4(3,500)
      DATA Q1,Q2,Q3,Q4/6000*1./
C
C***  INITIALIZE
C
      READ (5,1)
      READ (5,2) NPLOTS,PNLOCI,XSCALE,YSCALE,XPOSIT,YPOSIT
      PI=ATAN(1.)*4.
      GDIM = 3
      PNLOC = PNLOCI
C
      DO 10 L=1,NPLOTS
C
C***  GENERATE PLOT3D POINTER FILE
C
         DO 20 N=1,NPLOTS+1
            XLOC = N*2.*PI/NPLOTS
            SINY(1,N) = (0.0)*YSCALE + YPOSIT
            SINX(1,N) = (XLOC)*XSCALE + XPOSIT
            SINY(2,N) = (SIN(XLOC))*YSCALE + YPOSIT
            SINX(2,N) = (XLOC)*XSCALE + XPOSIT
20       CONTINUE
            PLOC = PNLOC*2.*PI/NPLOTS
            SINY(3,1) = (0.0)*YSCALE + YPOSIT
```

```
                  SINX(3,1) = (PLOC)*XSCALE + XPOSIT
                  SINY(3,2) = (SIN(PLOC))*YSCALE + YPOSIT
                  SINX(3,2) = (PLOC)*XSCALE + XPOSIT
C
             WRITE(50) GDIM,NPLOTS
             WRITE(50)((SINX(I,J), I=1,3),J=1,NPLOTS),
     +                ((SINY(I,J), I=1,3),J=1,NPLOTS)
C
             PNLOC = PNLOC + 1
             IF(PNLOC.EQ.NPLOTS)THEN
                  PNLOC = PNLOC + 1 - NPLOTS
             ENDIF
10      CONTINUE
C
C***    GENERATE DUMMY Q FILE
C
        WRITE(60) GDIM,NPLOTS
        WRITE(60) GDIM,GDIM,GDIM,GDIM
        WRITE(60) ((Q1(I,J), I=1,3),J=1,NPLOTS),
     +            ((Q2(I,J), I=1,3),J=1,NPLOTS),
     +            ((Q3(I,J), I=1,3),J=1,NPLOTS),
     +            ((Q4(I,J), I=1,3),J=1,NPLOTS)
C
1       FORMAT (1X)
2       FORMAT (1X,2I8,4F8.4)
        END
/EOF
 NPLOTS: PNLOCI: XSCALE: YSCALE: XPOSIT: YPOSIT:
 294     222     .08     .1      -1.0    -.5
```

5. CPPLOT.TXT

```
JOB,JN=SKYHAWK,T=30.  ERIC PAGENKOPF,X4269.
ACCOUNT,AC=,US=,UPW=.
*.
CFT,ON=A,OFF=S.
*.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=GETP3D,PDN=GETP3D,ID=RFRICC.
ACCESS,DN=SENDWKS,PDN=SENDWKS,ID=STTRDM,OWN=RFTRDM.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=CPX1,    PDN=SNKR05P,ID=FLYNAVY.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=CFX1,    PDN=SNKR05F,ID=FLYNAVY.
CSF,ACCESS,CSFACCT=,CSFPSWD=,DN=Q1,      PDN=SNKR05Q,ID=FLYNAVY.
REWIND,DN=CPX1.
REWIND,DN=CFX1.
REWIND,DN=Q1.
GETP3D,I=CPX1,O=CPX,DATATYPE=XYZ,DSSKIP=49,NUMDS=2.
GETP3D,I=CFX1,O=CFX,DATATYPE=XYZ,DSSKIP=49,NUMDS=2.
GETP3D,I=Q1,  O=Q,  DATATYPE=Q,  DSSKIP=49,NUMDS=2.
REWIND,DN=CPX.
REWIND,DN=CFX.
REWIND,DN=Q.
ASSIGN,DN=CPX,A=FT50.
ASSIGN,DN=CFX,A=FT60.
ASSIGN,DN=Q,  A=FT70.
*.
LDR,MAP=PART,SET=ZERO.
*.
*.
/EOF
      PROGRAM CPPLOT
*****************************************************************
*                                                               *
*     THIS PROGRAM READS STORED, TRUE VALUES OF XOC, CP AND CF, *
*     AND PRINTS THEM OUT TO TAPE 06.                           *
*                                                               *
*     VARIABLES:                                                *
*          NUMDS: NUMBER OF DATA SETS IN THE READ FILE          *
*          NPLOT: PLOT NO. OF FIRST DATA SET (DSSKIP + 1)       *
*                                                               *
*     TAPE ASSIGNMENTS:                                         *
*          TAPE 05: INPUT DATA                                  *
*          TAPE 06: OUTPUT DATA                                 *
*          TAPE 50: TRUE CP VALUES                              *
*          TAPE 60: TRUE CF VALUES                              *
*          TAPE 70: Q FILES                                     *
*                                                               *
*****************************************************************
C
      DIMENSION CPX(3,49),CFX(3,49),CPY(3,49),CFY(3,49)
      DIMENSION Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      DIMENSION KODE(4),LINE(90)
      DATA KODE/1H ,1H+,1HI,1H*/
C
C***  READ INPUT DATA & INITIALIZE
C
      READ (5,1)
      READ (5,2) NUMDS,NPLOT
      DO 9 I=1,90
         LINE(I) = KODE(1)
9     CONTINUE
C
C***  READ & SAVE TRUE VALUES
C
```

```
        DO 10 L1=1,NUMDS
             READ(50)   IMAX,KMAX
             READ(50)   ((CPX(I,J), I=1,IMAX),J=1,KMAX),
     +                  ((CPY(I,J), I=1,IMAX),J=1,KMAX)
             READ(60) IMAX,KMAX
             READ(60) ((CFX(I,J), I=1,IMAX),J=1,KMAX),
     +                ((CFY(I,J), I=1,IMAX),J=1,KMAX)
             READ(70)   IMAXQ,KMAXQ
             READ(70)   AMINF,ALFAD,REYREF,TIME
             READ(70)   ((Q1(I,J), I=1,IMAXQ),J=1,KMAXQ),
     +                  ((Q2(I,J), I=1,IMAXQ),J=1,KMAXQ),
     +                  ((Q3(I,J), I=1,IMAXQ),J=1,KMAXQ),
     +                  ((Q4(I,J), I=1,IMAXQ),J=1,KMAXQ)
             CP0 = (1. + .2 * AMINF**2) ** 3.5 - 1.
             CP0 = CP0 / ( .7 * AMINF**2)
             K0 = 30. * CP0 + 4.5
             DO 11 L=1,KMAX
                  K = 30. * (CP0 - CPY(3,L)) + 4.5
                  K1 = 30. * (CP0 -CPY(2,L)) + 4.5
                  IF(K.LT.1) K = 1
                  IF(K1.LT.1) K1 = 1
                  IF(K.GT.90) K = 90
                  IF(K1 .GT. 90) K1 = 90
                  LINE(K0) = KODE(3)
                  LINE(K) = KODE(2)
                  LINE(K1) = KODE(4)
                  IF(L.EQ.1)THEN
        WRITE(6,*)' PLOT    AOA       TIME      MACH     REY NO. '
        WRITE(6,3) NPLOT,ALFAD,TIME,AMINF,REYREF
        WRITE(6,*)' XOC      CFL      CFU      CPL      CPU '
        WRITE(6,4) CPX(1,L),CFY(2,L),CFY(3,L),CPY(2,L),CPY(3,L),LINE
                  ELSE
        WRITE(6,4) CPX(1,L),CFY(2,L),CFY(3,L),CPY(2,L),CPY(3,L),LINE
                  ENDIF
             LINE(K1)=KODE(1)
             LINE(K) =KODE(1)
11           CONTINUE
             NPLOT = NPLOT + 1
             WRITE(6,1)
10      CONTINUE
C
1       FORMAT (1X)
2       FORMAT (1X,2I7)
3       FORMAT (1X,I4,3F9.4,F10.0)
4       FORMAT (1X,F6.4,4F8.4,90A1)
C
        END
/EOF
 NUMDS: NPLOT:
 2      50
```

6. GETX.F

```
      PROGRAM GETX
C
***************************************************************
*                                                             *
*     THIS PROGRAM READS COMBINED (MULTIPLE DATA SET) FILES   *
*     OF XYZ AND Q PLOTTING DATA AND SEPARATES THEM INTO      *
*     INDIVIDUAL FILES FOR PLOT3D SUBMITTAL.                  *
*                                                             *
*     VARIABLES:                                              *
*          CFXNAME: NAME OF COMBINED XYZ FILE                 *
*          CFQNAME: NAME OF COMBINED Q FILE                   *
*          FXNAME: NAME OF INDIVIDUAL X FILES, CHAR VARIABLE  *
*          FQNAME: NAME OF INDIVIDUAL Q FILES, CHAR VARIABLE  *
*          DSSKIP: NUMBER OF DATA SETS TO SKIP WHEN           *
*                  READING COMBINED FILE                      *
*          DSREAD: NUMBER OF DATA SETS TO READ                *
*          INTVL:  INTERVAL BETWEEN DATA SETS SENT TO FILE    *
*                                                             *
***************************************************************
C
      INTEGER DSSKIP,DSREAD,COUNT
      CHARACTER FXNAME*12,FQNAME*12,CFXNAME*10,CFQNAME*10
      DIMENSION X(250,100),Z(250,100)
      DIMENSION Q1(250,100),Q2(250,100),Q3(250,100),Q4(250,100)
C
*** USER INPUT: ********************************************
C
      DATA XSCALE,YSCALE/1.,1./
      DATA XPOSIT,YPOSIT/-1.,-1./
      DATA DSSKIP/10/,DSREAD/3/,INTVL/1/
      CFXNAME = '38x.iris'
      CFQNAME = '38q.iris'
C
***********************************************************
C
      OPEN(UNIT=91,FILE=CFXNAME,STATUS='OLD',FORM='BINARY')
      OPEN(UNIT=92,FILE=CFQNAME,STATUS='OLD',FORM='BINARY')
C
      FXNAME = 'x000.iris'
      FQNAME = 'q000.iris'
      COUNT  =  0
C
      IF (DSSKIP.GT.0) THEN
           DO 10 ISKIP = 1,DSSKIP
                READ(91) IMAX,KMAX
                READ(91) ((X(I,K),I=1,IMAX),K=1,KMAX),
     +                   ((Z(I,K),I=1,IMAX),K=1,KMAX)
                READ(92) IMAX,KMAX
                READ(92) A,B,C,D
                READ(92) ((Q1(I,K),I=1,IMAX),K=1,KMAX),
     +                   ((Q2(I,K),I=1,IMAX),K=1,KMAX),
     +                   ((Q3(I,K),I=1,IMAX),K=1,KMAX),
     +                   ((Q4(I,K),I=1,IMAX),K=1,KMAX)
10         CONTINUE
      ENDIF
C
      DO 20 IREAD = 1,DSREAD
           COUNT = COUNT + 1
           WRITE (FXNAME(2:4),100) IREAD
           WRITE (FQNAME(2:4),100) IREAD
           READ(91) IMAX,KMAX
           READ(91) ((X(I,K),I=1,IMAX),K=1,KMAX),
     +              ((Z(I,K),I=1,IMAX),K=1,KMAX)
```

```
          IF (COUNT.EQ.INTVL) THEN
               OPEN(UNIT=1,FILE=FXNAME,FORM='BINARY')
               DO 11 I=1,IMAX
                    DO 12 K=1,KMAX
                         X(I,K)=X(I,K)*XSCALE+XPOSIT
                         Z(I,K)=Z(I,K)*YSCALE+YPOSIT
12                  CONTINUE
11             CONTINUE
               WRITE(1) IMAX,KMAX
               WRITE(1) ((X(I,K),I=1,IMAX),K=1,KMAX),
     +                  ((Z(I,K),I=1,IMAX),K=1,KMAX)
               CLOSE(1)
          ENDIF
          READ(92) IMAX,KMAX
          READ(92) A,B,C,D
          READ(92) ((Q1(I,K),I=1,IMAX),K=1,KMAX),
     +             ((Q2(I,K),I=1,IMAX),K=1,KMAX),
     +             ((Q3(I,K),I=1,IMAX),K=1,KMAX),
     +             ((Q4(I,K),I=1,IMAX),K=1,KMAX)
          IF (COUNT.EQ.INTVL) THEN
               OPEN(UNIT=2,FILE=FQNAME,FORM='BINARY')
               WRITE(2) IMAX,KMAX
               WRITE(2) A,B,C,D
               WRITE(2) ((Q1(I,K),I=1,IMAX),K=1,KMAX),
     +                  ((Q2(I,K),I=1,IMAX),K=1,KMAX),
     +                  ((Q3(I,K),I=1,IMAX),K=1,KMAX),
     +                  ((Q4(I,K),I=1,IMAX),K=1,KMAX)
               CLOSE(2)
               COUNT = 0
          ENDIF
20    CONTINUE
C
      CLOSE(91)
      CLOSE(92)
100   FORMAT(I3.3)
      STOP
      END
```

7. GETCP.F

```
      PROGRAM GETCP
C
*****************************************************************
*                                                               *
*     THIS PROGRAM ACCESSES COMBINED CP OR CF FILES ON THE      *
*     IRIS ACCOUNT, SEPARATES THEM INTO INDIVIDUAL PLOTTING     *
*     FILES AND SCALES THEM TO MEET SPECIFIED REQUIREMENTS.     *
*     A DUMMY Q FILE IS ALSO GENERATED (Q.IRIS).                *
*                                                               *
*     VARIABLES:                                                *
*                                                               *
*          CPNAME: COMBINED FILE NAME                           *
*          FNAME: OUTPUT FILE NAME, CHARACTER VARIABLE          *
*          DSSKIP: NUMBER OF DATASETS IN THE COMBINED FILE      *
*                  TO SKIP                                      *
*          DSFILE: NUMBER OF DATASETS IN THE COMBINED FILE      *
*                  TO SEPARATE AND FILE                         *
*          XSCALE: AXIS SCALING FACTOR                          *
*          YSCALE: CP/CF SCALING FACTOR (RELATIVE MAGNITUDE)    *
*          XPOSIT: X POSITION FACTOR (SCREEN LOCATION)          *
*          YPOSIT: Y POSITION FACTOR (SCREEN LOCATION)          *
*                                                               *
*****************************************************************
C
      INTEGER DSSKIP,DSFILE
      CHARACTER FNAME*11,QNAME*6,CFNAME*10
      DIMENSION X(3,75),Z(3,75),XS(3,75),ZS(3,75)
      DIMENSION Q1(3,75),Q2(3,75),Q3(3,75),Q4(3,75)
      DATA Q1,Q2,Q3,Q4/900*1./
C
*** USER INPUTS: **************************************
C
      DATA DSSKIP/250/,DSFILE/44/
      DATA XSCALE/.5/,YSCALE/-.05/
      DATA XPOSIT/-1.0/,YPOSIT/.5/
      CFNAME = 'snkr05p'
      FNAME = 'cp000.iris'
C
*****************************************************
C
      OPEN(UNIT=90,FILE=CFNAME,STATUS='OLD',FORM='BINARY')
C
      QNAME = 'q.iris'
C
***  SKIP DSSKIP FILES
C
      IF (DSSKIP.GT.0) THEN
           DO 10 ISKIP = 1,DSSKIP
           READ(90) IMAX,KMAX
           READ(90) ((X(I,K),I=1,IMAX),K=1,KMAX),
    +               ((Z(I,K),I=1,IMAX),K=1,KMAX)
10         CONTINUE
      ENDIF
C
***  SEPARATE AND SCALE DSFILE FILES
C
      DO 40 IFILE = 1,DSFILE
           WRITE(FNAME(3:5),100) IFILE
           OPEN(UNIT=1,FILE=FNAME,FORM='BINARY')
           READ(90) IMAX,KMAX
           READ(90) ((X(I,K),I=1,IMAX),K=1,KMAX),
    +               ((Z(I,K),I=1,IMAX),K=1,KMAX)
```

```
            DO 30 I = 1,IMAX
                  DO 20 K = 1,KMAX
                        XS(I,K)=X(I,K)*XSCALE + XPOSIT
                        ZS(I,K)=Z(I,K)*YSCALE + YPOSIT
20                CONTINUE
30          CONTINUE
            WRITE(1) IMAX,KMAX
            WRITE(1) ((XS(I,K),I=1,IMAX),K=1,KMAX),
     +               ((ZS(I,K),I=1,IMAX),K=1,KMAX)
            CLOSE(1)
40    CONTINUE
C
***   GENERATE DUMMY Q FILE
C
      OPEN(UNIT=1,FILE=QNAME,FORM='BINARY')
      WRITE(1) IMAX,KMAX
      WRITE(1) IMAX,IMAX,IMAX,IMAX
      WRITE(1) ((Q1(I,K),I=1,IMAX),K=1,KMAX),
     +         ((Q2(I,K),I=1,IMAX),K=1,KMAX),
     +         ((Q3(I,K),I=1,IMAX),K=1,KMAX),
     +         ((Q4(I,K),I=1,IMAX),K=1,KMAX)
      CLOSE(1)
C
100   FORMAT(I3.3)
C
      STOP
      END
```

8. GETSIN.F

```
      PROGRAM GETSIN
C
*****************************************************************
*                                                               *
*     THIS PROGRAM READS A COMBINED FILE OF AOA POINTER         *
*     DATA (GENERATED WITH PLOT3D FUNCTION 0) AND SEPARATES     *
*     IT INTO INDIVIDUAL PLOTTING FILES.                        *
*                                                               *
*     VARIABLES:                                                *
*          CFNAME: NAME OF COMBINED FILE                        *
*          FNAME: NAME OF INDIVIDUAL FILES, CHAR VARIABLE       *
*          DSSKIP: NUMBER OF DATA SETS TO SKIP WHEN             *
*                  READING COMBINED FILE                        *
*          DSFILE: NUMBER OF DATA SETS TO SEND TO INDIVIDUAL    *
*                  FILES                                        *
*                                                               *
*****************************************************************
C
      INTEGER DSSKIP,DSFILE
      CHARACTER FNAME*12,CFNAME*10
      DIMENSION X(3,294),Z(3,294)
C
*** USER INPUT: *********************************************
C
      DATA DSSKIP/250/,DSFILE/44/
      CFNAME = 'snkr05s'
C
*************************************************************
C
      OPEN(UNIT=90,FILE=CFNAME,STATUS='OLD',FORM='BINARY')
C
      FNAME = 'sin000.iris'
C
      IF (DSSKIP.GT.0) THEN
           DO 10 ISKIP = 1,DSSKIP
                READ(90) IMAX,KMAX
                READ(90) ((X(I,J),I=1,IMAX),J=1,KMAX),
     +                   ((Z(I,J),I=1,IMAX),J=1,KMAX)
10         CONTINUE
      ENDIF
C
      DO 20 IFILE = 1,DSFILE
           WRITE (FNAME(4:6),100) IFILE
           OPEN(UNIT=1,FILE=FNAME,FORM='BINARY')
           READ(90) IMAX,KMAX
           READ(90) ((X(I,J),I=1,IMAX),J=1,KMAX),
     +              ((Z(I,J),I=1,IMAX),J=1,KMAX)
           WRITE(1) IMAX,KMAX
           WRITE(1) ((X(I,J),I=1,IMAX),J=1,KMAX),
     +             ((Z(I,J),I=1,IMAX),J=1,KMAX)
           CLOSE(1)
20    CONTINUE
C
      CLOSE(90)
100   FORMAT(I3.3)
      STOP
      END
```

C. PROCEDURAL DOCUMENTATION

After proper initialization (user-inputs) as described above, all codes or JCL's which produce or transfer Plot3D files are submitted to the Cray utilizing standard CSUB commands. Once the data is resident on the IRIS and the user is logged on and in the proper directory level, the following procedures may be utilized for graphics generation.

1. Separate the combined file into discrete data sets.

   Edit "user inputs" section of GETX.F (or other files as appropriate) to define scaling (ultimate location on the display), and identify the combined file to access and those data sets to be separated:

   vi getx.f

   Compile the editted program:

   f77 getx.f

   Run the compiled version by entering the filename (a.out by default).

   a.out

   Discrete data sets will appear on the account with names corresponding to those in the Plot3D initialization files. Check by using the ls command.

2. Use Plot3D to generate *.gra* files.

   Use the mv commands to give the proper initialization file the filename "plot3dini.com". (The IRIS will not save multiple versions of files with the same filename. Instead, older versions of the file will be deleted. In order to avoid inadvertent deletion of initialization files, multiple mv commands may be required.)

   mv xini.com plot3dini.com

   If processing flow field files (X and Q), edit the initialization file to identify desired function and number of contours. This involves changing only two lines at the top of the file.

   vi plot3dini.com

   Create *.gra* files.

   plot3x

   Rename the resultant combined graphics file.

   mv q001.gra somefilnm.gra

   After creation of as many *.gra* files as required from the separated files, clean up the account using the following wildcards.

   rm q*.iris

   rm x*.iris

3. Interactive viewing.

Run the Graphics Animation System software.

gas

Input *.gra* files to GAS by selecting on the main menu:

ARCGRAPH file input

On the submenu, select:

load entire file

In response to prompts, enter the sequential object number and file name ("someflnm.gra"). For color plots, press RETURN when prompted for the colormap.

View the data by selecting on the main menu:

view data

Utilize the mouse to manipulate the display.

APPENDIX B. SANKAR NAVIE STOKES SOLVER

```
(JCL)
*.
/EOF
      PROGRAM MAIN
************************************************************************
*                                                                      *
*     SANKAR NAVIER STOKES CODE:  SOLVES TWO DIMENSIONAL FLOWS PAST    *
*     ARBITRARY GEOMETRIES USING ADI PROCEDURE.  THIS VERSION SAVES    *
*     MULTIPLE PLOT3D DATA SETS IN SINGLE FILES.                       *
*                                                                      *
*     VARIABLES:                                                       *
*          TITLE: (60 CHARACTERS MAX.)                                 *
*          IMAX: NUMBER OF X COORD LOCATIONS                           *
*          KMAX: NUMBER OF Y COORD LOCATIONS                           *
*          ITEL: GRID POINT AT LOWER TRAILING EDGE                     *
*          ITEU: GRID POINT AT UPPER TRAILING EDGE                     *
*          DT: SIZE OF TIME STEP                                       *
*          WW: EXPLICIT ARTIFICIAL VISCOSITY TERM                      *
*          ALFA: MEAN ANGLE OF ATTACK (DEGREES)                        *
*          ALFA1: AMPLITUDE OF OSCILLATION (DEGREES)                   *
*          ALFAI: ANGLE BELOW WHICH MODIFIED TURBULENCE MODEL USED TO  *
*                 COMPUTE EDDY VISCOSITY                               *
*          REDFRE: REDUCED FREQUENCY                                   *
*          AMINF: FREE STREAM MACH NUMBER                              *
*          REYREF: REYNOLDS NUMBER (MILLIONS; NEG. = INVISCID FLOW)    *
*          DNMIN: DISTANCE OF FIRST POINT OFF OF WALL                  *
*          TSTART: TIME FOR CALCULATIONS TO START ON PRESENT RUN       *
*          FORMAT: OLDSLN FORMAT;3.0=PLOT3D FILES,-3.0=Q MATRICES      *
*          RSTRT: TRUE = STORED DATA READ TO CONTINUE ITERATION        *
*          PITCH: TRUE = AIRFOL AOA OSCILLATES                         *
*          PLUNGE: TRUE = AIRFOIL OSCILLATES VERTICALLY                *
*          FNU: NUMBER OF UPPER SURFACE DEFINITION POINTS              *
*          FNL: NUMBER OF LOWER SURFACE DEFINITION POINTS              *
*          FSYM: SYMMETRY FLAG (1 = SYMMETRIC)                         *
*          X: AIRFOIL GEOMETRY DEFINITION POINTS                       *
*          Y: AIRFOIL GEOMETRY DEFINITION POINTS                       *
*          CSTP: NUMBER OF STEPS COMPLETED                             *
*          CPLT: NUMBER OF DYNAMIC PLOTS COMPLETED                     *
*          NSTP: NUMBER OF TIME STEPS TO BE COMPLETED ON PRESENT RUN   *
*          PSTP: NUMBER OF TIME STEPS BETWEEN PLOT DATA OUTPUT         *
*                                                                      *
*     TAPE DEFINITIONS:                                                *
*          TAPE 05: INPUT DATA                                         *
*          TAPE 06: OUTPUT DATA                                        *
*          TAPE 07: FLOW FIELD INPUT DATA FOR RESTARTS                 *
*          TAPE 20: PLOT3D XYZ FILE STORAGE                            *
*          TAPE 21: PLOT3D Q FILE STORAGE                              *
*          TAPE 50: PLOT3D CP XYZ FILE STORAGE                         *
*          TAPE 60: PLOT3D CF XYZ FILE STORAGE                         *
*          TAPE 90: PREVIOUS RUN X FILE STORAGE                        *
*          TAPE 91: PREVIOUS RUN Q FILE STORAGE                        *
*          TAPE 92: PREVIOUS RUN CP FILE STORAGE                       *
*          TAPE 93: PREVIOUS RUN CF FILE STORAGE                       *
*                                                                      *
************************************************************************
C
      INTEGER PSTP,CSTP,CPLT
```

```
      COMMON/SURF/PSUR(161)
      COMMON/FIX/OMEGA
      COMMON/MUTUR/CMU(161,41)
      COMMON/SKINCF/CF(161)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      DIMENSION TITLE(15),TYTLE(15),ALPHA(96),CPTH(97,96),XTH(97)
      DIMENSION XR(161,49),ZR(161,49),Q1R(161,41),Q2R(161,41)
      DIMENSION Q3R(161,41),Q4R(161,41)
      COMMON/LOGIC/RSTRT,PITCH,PLUNGE
      LOGICAL RSTRT,PITCH,PLUNGE
C
C***  READ INPUT DATA
C
      READ (5,5)
      READ (5,1) TITLE
      READ (5,5)
      READ (5,2) IMAX,KMAX,DT,WW,ALFA,ALFA1,ALFAI,REDFRE,AMINF
      READ (5,5)
      READ (5,3) ITEL,ITEU,REYREF,DNMIN,TSTART,FORMAT,RSTRT,PITCH,PLUNGE
      READ (5,5)
      READ (5,4) CSTP,CPLT,NSTP,PSTP
C
C***  INITIALIZE
C
      GAMMA = 1.4
      PI = ATAN(1.)*4.
C     SET ALFAD FOR STEADY STATE PLOT3D OUTPUT
      ALFAD = ALFA
C     FORCE DT TO BE EQUAL TO UNITY FOR STEADY FLOW PROBLEMS
C     THIS INVOKES THE RELAXATION MODE
      IF(REDFRE.LE.0.001) DT = 1.0
      REYREF = REYREF * 1.E+06
      NITN = CSTP + NSTP
      NPLOTS = CPLT
      CPLT = CPLT + 1
      CSTP = CSTP + 1
C     DENSITY NORMALISED WITH RESPECT TO ROINF
C     VELOCITIES NORMALISED WITH RESPECT TO AINF
C     TOTAL ENERGY NORMALISED WITH RESPECT TO (ROINF*AINF*AINF)
      TOTEN=AMINF*AMINF*0.5+1./(GAMMA*(GAMMA-1.))
      ALFA = ALFA * ATAN(1.) / 45.
      ALMEAN = ALFA
      ALFAI = ALFAI * ATAN(1.) / 45.
      ALFA1 = ALFA1 * ATAN(1.) / 45.
      ALFAS = ALMEAN - ALFA1
      WRITE(6)' PLOT  ITN    TIME    AOA      CL      CD      CM'
      UINF = AMINF
      VINF = 0.0
      DO 7 I=1,IMAX
      DO 7 K=1,KMAX
         Q1(I,K)=1.
         Q2(I,K)=UINF
         Q3(I,K)=VINF
         Q4(I,K)=TOTEN
    7 CONTINUE
C
C***  INPUT STORED FLOW FIELD DATA
C
      IF(RSTRT)THEN
           IF(FORMAT.LT.0.0)THEN
                READ(7) TIME,Q1,Q2,Q3,Q4
           ELSE
                READ(7) IMAX,KMAX
```

```
                READ(7) AMINF,ALFAD,REYREF,TIME
                READ(7) ((Q1(I,J), I=1,IMAX),J=1,KMAX),
     +                  ((Q2(I,J), I=1,IMAX),J=1,KMAX),
     +                  ((Q3(I,J), I=1,IMAX),J=1,KMAX),
     +                  ((Q4(I,J), I=1,IMAX),J=1,KMAX)
            ENDIF
            IF(NPLOTS.GT.0)THEN
                DO 9 K1=1,NPLOTS
                    READ(90)  IMAXR,KMAXR
                    READ(90)  ((XR(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((ZR(I,J), I=1,IMAXR),J=1,KMAXR)
                    WRITE(20) IMAXR,KMAXR
                    WRITE(20) ((XR(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((ZR(I,J), I=1,IMAXR),J=1,KMAXR)
 9              CONTINUE
                DO 10 K2=1,NPLOTS
                    READ(91)  IMAXR,KMAXR
                    READ(91)  AMINFR,ALFADR,REYREFR,TIMER
                    READ(91)  ((Q1R(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((Q2R(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((Q3R(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((Q4R(I,J), I=1,IMAXR),J=1,KMAXR)
                    WRITE(21) IMAXR,KMAXR
                    WRITE(21) AMINFR,ALFADR,REYREFR,TIMER
                    WRITE(21) ((Q1R(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((Q2R(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((Q3R(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((Q4R(I,J), I=1,IMAXR),J=1,KMAXR)
 10             CONTINUE
                DO 11 K3=1,NPLOTS
                    READ(92)  IMAXR,KMAXR
                    READ(92)  ((XR(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((ZR(I,J), I=1,IMAXR),J=1,KMAXR)
                    WRITE(50) IMAXR,KMAXR
                    WRITE(50) ((XR(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((ZR(I,J), I=1,IMAXR),J=1,KMAXR)
 11             CONTINUE
                DO 12 K4=1,NPLOTS
                    READ(93)  IMAXR,KMAXR
                    READ(93)  ((XR(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((ZR(I,J), I=1,IMAXR),J=1,KMAXR)
                    WRITE(60) IMAXR,KMAXR
                    WRITE(60) ((XR(I,J), I=1,IMAXR),J=1,KMAXR),
     +                        ((ZR(I,J), I=1,IMAXR),J=1,KMAXR)
 12             CONTINUE
            ENDIF
        ENDIF
        IF(TSTART.GE.0.) TIME = TSTART
        IF(.NOT.(RSTRT)) TIME = 0.
C
C***  GENERATE COMPUTATIONAL GRID, DEFINE SURFACE COORD @ 0 AOA,
C      ROTATE TO INITIAL AOA AND COMPUTE METRICS
C
      CALL AIRFOL(IMAX,KMAX,ITEL,ITEU)
      IF (REYREF.GT.0) CALL CLUSTR(DNMIN)
      TEDGE = X(ITEL + 48,1)
      DO 15 I = 1,97
         XTH(I) = X(I+ITEL-1,1)-TEDGE
 15   CONTINUE
      CALL ROTGRID(X,Z,IMAX,KMAX,ALFAS)
      CALL METRIC
C
C*** ITERATIVE LOOP
C
      DO 1000 ITN=1,NSTP
         TIME = TIME + DT
C
```

```
C***  ROTATE GRID TO NEW ANGLE OR SET TO CORRECT ANGLE FOR RESTARTS
C
         IF (PITCH) THEN
            OMEGA = 2. * REDFRE *AMINF*SIN(REDFRE * 2. * TIME * AMINF)
     1      *ALFA1
            ALOLD = ALMEAN - ALFA1 * COS(2. * REDFRE * AMINF *
     1      (TIME - DT))
            ALFA = ALMEAN - ALFA1 * COS(REDFRE * 2. * TIME * AMINF)
            ALFAD = ALFA * 45. / ATAN(1.)
            DALFA = ALFA - ALOLD
            IF(RSTRT.AND.ITN.EQ.1) DALFA = ALFA - 2.*ALFAS
            CALL ROTGRID(X,Z,IMAX,KMAX,DALFA)
            CALL METRIC
         END IF
         IF (PLUNGE) THEN
            OMEGA = 2. * REDFRE * AMINF
            ALFA = ALMEAN
         END IF
C
C***  COMPUTE THE SOLUTION BY ADI TECHNIQUE
C
         CALL SLPS(ITN,OMEGA,DALFA)
C
C***  APPLY BOUNDARY CONDITIONS
C
         CALL WALLBC
C
C***  GENERATE PLOT3D FILES
C
      IF(CSTP.EQ.(CPLT*PSTP)) THEN
           WRITE(20) IMAX, KMAX
           WRITE(20) ((X(I,J), I=1,IMAX), J=1,KMAX),
     1               ((Z(I,J), I=1,IMAX), J=1,KMAX)
           WRITE(21) IMAX, KMAX
           WRITE(21) AMINF, ALFAD, REYREF, TIME
           WRITE(21) ((Q1(I,J), I=1,IMAX), J=1,KMAX),
     1                  ((Q2(I,J), I=1,IMAX), J=1,KMAX),
     2                  ((Q3(I,J), I=1,IMAX), J=1,KMAX),
     3                  ((Q4(I,J), I=1,IMAX), J=1,KMAX)
C
C***  GENERATE PERFORMANCE COEFFICIENTS
C
           CALL LOAD(PSUR,CL,CD,CM,ALFAS)
           AOA = ALFA*180./PI
           WRITE(6,6) CPLT,CSTP,TIME,AOA,CL,CD,CM
           CALL CPPLOT(ITEL,ITEU,AMINF,X(1,1),Z(1,1),PSUR)
           CPLT = CPLT + 1
      END IF
      CSTP = CSTP + 1
1000  CONTINUE
C
1     FORMAT (1X,15A4)
2     FORMAT (1X,2I7,7F8.4)
3     FORMAT (1X,2I7,4F8.5,3L7)
4     FORMAT (1X,4I7)
5     FORMAT (1X)
6     FORMAT (1X,I3,I6,F8.3,F9.5,3F8.4)
C
      END
C
C*************************************************************************
C
      SUBROUTINE AMAT1(K,IMX1,XIX,XIZ,XIT)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/AM/A(4,4,161)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
```

```
      DIMENSION XIT(161,41),XIX(161,41),XIZ(161,41)
      REAL K0,K1,K2
C
C***  AMAT1 COMPUTES THE COEFFICIENT MATRIX DE/DQ DURING XI SWEEP
C
      GM1          = GAMMA - 1.
      DO 1000 I = 2 , IMX1
      K0           = XIT(I,K)
      K1           = XIX(I,K)
      K2           = XIZ(I,K)
      U            = Q2(I,K) / Q1(I,K)
      W            = Q3(I,K) / Q1(I,K)
      EBYR         = Q4(I,K) / Q1(I,K)
      PHI2         = 0.5 * GM1 * (U * U + W * W)
      THETA        = K1 * U + K2 * W
      A(1,1,I)     = K0
      A(1,2,I)     = K1
      A(1,3,I)     = K2
      A(1,4,I)     = 0
      A(2,1,I)     = K1 * PHI2 - U * THETA
      A(2,2        = K0 + THETA - K1 * (GM1 - 1.) * U
      A(2,3          K2 * U - GM1 * K1 * W
      A(2,4          K1 * GM1
      A(3,           K2 * PHI2 - W * THETA
      A(3,           K1 * W - K2 * GM1 * U
      A(3,           K0 + THETA - K2 * (GM1 - 1.) * W
      A(3,         = K2 * GM1
      A(4,         = THETA * (2. * PHI2 - GAMMA * EBYR)
      A(4,         = K1 * (GAMMA * EBYR - PHI2) - GM1 * U * THETA
      A(4,3,       = K2 * (GAMMA * EBYR - PHI2) - GM1 * W * THETA
      A(4,4,       = K0 + GAMMA * THETA
 1000 CONTINUE
      RETURN
      END
C
C*****************************************************************************
C
      SUBROUTINE AMAT2(I,KMX1,ZETAX,ZETAZ,ZETAT)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/AM/A(4,4,161)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      DIMENSION ZETAX(161,41),ZETAZ(161,41),ZETAT(161,41)
      REAL K0,K1,K2
C
C***  AMAT2 COMPUTES THE COEFFICIENT MATRIX DF/DQ DURING ETA SWEEP
C
      GM1          = GAMMA - 1.
      DO 1000 K = 2 , KMX1
      K0           = ZETAT(I,K)
      K1           = ZETAX(I,K)
      K2           = ZETAZ(I,K)
      U            = Q2(I,K) / Q1(I,K)
      W            = Q3(I,K) / Q1(I,K)
      EBYR         = Q4(I,K) / Q1(I,K)
      PHI2         = 0.5 * GM1 * (U * U + W * W)
      THETA        = K1 * U + K2 * W
      A(1,1,K)     = K0
      A(1,2,K)     = K1
      A(1,3,K)     = K2
      A(1,4,K)     = 0
      A(2,1,K)     = K1 * PHI2 -U * THETA
      A(2,2,K)     = K0 + THETA - K1 * (GM1-1.) * U
      A(2,3,K)     = K2 * U - GM1 * K1 * W
      A(2,4,K)     = K1 * GM1
      A(3,1,K)     = K2 * PHI2 - W * THETA
      A(3,2,K)     = K1 * W - K2 * GM1 * U
      A(3,3,K)     = K0 + THETA -K2 * (GM1-1.) * W
```

```
      A(3,4,K)   = K2 * GM1
      A(4,1,K)   = THETA * (2. * PHI2 - GAMMA * EBYR)
      A(4,2,K)   = K1 * (GAMMA * EBYR - PHI2) - GM1 * U * THETA
      A(4,3,K)   = K2 * (GAMMA * EBYR - PHI2) - GM1 * W * THETA
      A(4,4,K)   = K0 + GAMMA * THETA
 1000 CONTINUE
      RETURN
      END
C
C**************************************************************************
C
      SUBROUTINE SLPS(ITN,OMEGA,DALFA)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/AM/A(4,4,161)
      COMMON/TRID/DD(4,4,161,41),MM(4,4,161,41),EE(4,4,161,41)
     1,GG(4,161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      COMMON/MTRIX/ XIX(161,41),XIZ(161,41),ZETAX(161,41),
     +ZETAZ(161,41)
     1 ,XIT(161,41),ZETAT(161,41)
      REAL MM
      DIMENSION DELTAT(161,41)
C
C***  SUBROUTINE SLPS DOES THE BULK OF THE WORK  FOR THE AC: ALGORITHM.
C***  IT CALLS FLUX AND COMPUTES RIGHT HAND SIDE DURING THE TWO SWEEPS,
C***  ASSEMBLES THE COEFFICIENT MARICES, ADDS IMPLICIT AND EXPLICIT
C***  DISSIPATION AND CALLS THE TRIDIAGONAL SOLVER TO OBTAIN THE FINAL
C***  SOLUTION.
C!!!!!SET VALUE OF IMPLICIT DAMPING COEFFICIENT
      WWI = 20.0 * WW
      IM1 = IMAX - 1
      IM2 = IMAX - 2
      KM1 = KMAX - 1
      KM2 = KMAX - 2
      IF(ITN.EQ.1) THEN
C!!!!!LOCAL TIME STEP OPTION FOR STEADY STATE CALCULATIONS
      IF(REDFRE.LT.0.001) THEN
      DO 777 K = 2 , KMAX - 1
      DO 777 I = 2 , IMAX - 1
      DELTAT(I,K) = 0.5 / (1. + SQRT(ABS(YACOB(I,K))))
  777 CONTINUE
      ELSE
      DO 778 K = 2 , KMAX - 1
      DO 778 I = 2 , IMAX - 1
  778 DELTAT(I,K) = 1.0
      END IF
      END IF
C
C***  THE DISSIPATION TERMS ARE CONSTRUCTED AND STORED IN THE ARRAYS DQ1,
C***  DQ2,DQ3 AND DQ4.
C
      CALL DISSIP
C
C***  THE RIGHT HAND SIDE AT KNOWN TIME LEVEL IS NOW COMPUTED AND ADDED
      CALL RESI(OMEGA)
C
C***  IF VISCOUS FLOW IS COMPUTED THE VISCOUS TERMS ARE ADDED TO DQ1 ETC. HERE.
C
      IF(REYREF.GT.0.) CALL STRESS(ITN,DALFA)
C***  I-SWEEP.
C
      DTH = DT * 0.5
      DTW = DT * WWI
```

```
      DO 3 K            = 2 , KM1
      CALL AMAT1(K,IMAX-1,XIX,XIZ,XIT)
      DO 4 I1           = 1 , 4
      DO 4 I2           = 1 , 4
      DO 5 I            = 2 , IMAX - 1
      EE(I1,I2,I-1,K) =   A(I1,I2,I+1) * DTH * DELTAT(I,K)
      DD(I1,I2,I-1,K) = - A(I1,I2,I-1) * DTH * DELTAT(I,K)
    5 CONTINUE
    4 CONTINUE
C
C***  IMPLICIT DAMPING ADDED HERE.
C
      DO 6 I1           = 1 , 4
      DO 6 I            = 2 , IMAX - 1
      DT1               = DTW / YACOB(I,K) * DELTAT(I,K)
      DD(I1,I1,I-1,K) = DD(I1,I1,I-1,K) - DT1 * YACOB(I-1,K)
      EE(I1,I1,I-1,K) = EE(I1,I1,I-1,K) - DT1 * YACOB(I+1,K)
      MM(I1,I1,I-1,K) = 1. + 2. * DTW * DELTAT(I,K)
    6 CONTINUE
      DO 990 I          = 2 , IMAX - 1
      GG(1,I-1,K)       = DQ1(I,K) * DELTAT(I,K)
      GG(2,I-1,K)       = DQ2(I,K) * DELTAT(I,K)
      GG(3,I-1,K)       = DQ3(I,K) * DELTAT(I,K)
      GG(4,I-1,K)       = DQ4(I,K) * DELTAT(I,K)
  990 CONTINUE
    3 CONTINUE
C
C     PERFORM BLOCK TRIDIAGONAL MATRIX INVERSION FOR THE ENTIRE PLANE
C
      CALL MATRX1(IMAX,KMAX)
      DO 991 K          = 2 , KMAX - 1
      DO 991 I          = 2 , IM1
      DQ1(I,K)          = GG(1,I-1,K)
      DQ2(I,K)          = GG(2,I-1,K)
      DQ3(I,K)          = GG(3,I-1,K)
      DQ4(I,K)          = GG(4,I-1,K)
  991 CONTINUE
C
C     K-SWEEP BEGINS HERE.
C
      DO 13 I           = 2 , IM1
      CALL AMAT2(I,KMAX-1,ZETAX,ZETAZ,ZETAT)
      DO 15 I1          = 1 , 4
      DO 15 I2          = 1 , 4
      DO 15 K           = 2 , KMAX - 1
      EE(I1,I2,I,K-1) = A(I1,I2,K+1)*DTH * DELTAT(I,K)
      DD(I1,I2,I,K-1) = -A(I1,I2,K-1)*DTH * DELTAT(I,K)
   15 CONTINUE
C
C***  SECOND ORDER DAMPING ADDED HERE.
C
      DO 16 I1          = 1 , 4
      DO 16 K           = 2 , KMAX - 1
      DT1               = DTW / YACOB(I,K) * DELTAT(I,K)
      DD(I1,I1,I,K-1) = DD(I1,I1,I,K-1) - DT1 * YACOB(I,K-1)
      EE(I1,I1,I,K-1) = EE(I1,I1,I,K-1) - DT1 * YACOB(I,K+1)
   16 MM(I1,I1,I,K-1) = 1. + 2. *  DTW * DELTAT(I,K)
      DO 17 K           = 2 , KMAX - 1
      GG(1,I,K-1)       = DQ1(I,K)
      GG(2,I,K-1)       = DQ2(I,K)
      GG(3,I,K-1)       = DQ3(I,K)
      GG(4,I,K-1)       = DQ4(I,K)
   17 CONTINUE
   13 CONTINUE
C
C     PERFORM BLOCK TRIDIAGONAL MATRIX INVERSION FOR THE ENTIRE PLANE
C
```

```
      CALL MATRX2(IMAX,KMAX)
      DO 18 I          = 2 , IMAX - 1
      DO 18 K          = 2 , KM1
      DQ1(I,K)         = GG(1,I,K-1)
      DQ2(I,K)         = GG(2,I,K-1)
      DQ3(I,K)         = GG(3,I,K-1)
      DQ4(I,K)         = GG(4,I,K-1)
   18 CONTINUE
C
C***  UPDATE FLOW VARIABLES AT INTERIOR POINTS.
  967 CONTINUE
      RMAX             = 0.
      RUMAX            = 0.
      RVMAX            = 0.
      EMAX             = 0.
      DO 995 K         = 2 , KM1
      DO 19  I         = 2 , IM1
      Q1(I,K)          = Q1(I,K) + DQ1(I,K) * YACOB(I,K)
      Q2(I,K)          = Q2(I,K) + DQ2(I,K) * YACOB(I,K)
      Q3(I,K)          = Q3(I,K) + DQ3(I,K) * YACOB(I,K)
      Q4(I,K)          = Q4(I,K) + DQ4(I,K) * YACOB(I,K)
   19 CONTINUE
C!!!!!DETERMINE WHERE IN FLOW FIELD DENSITY IS CHANGING MOST RAPIDLY
      DO 995 I         = 2 , IMAX - 1
      IF (RMAX.LT.ABS(DQ1(I,K)*YACOB(I,K))) THEN
      IR               = I
      KR               = K
      END IF
      RMAX             = AMAX1(RMAX,ABS(DQ1(I,K) * YACOB(I,K)))
      RUMAX            = AMAX1(RUMAX,ABS(DQ2(I,K) * YACOB(I,K)))
      RVMAX            = AMAX1(RVMAX,ABS(DQ3(I,K) * YACOB(I,K)))
      EMAX             = AMAX1(EMAX,ABS(DQ4(I,K) * YACOB(I,K)))
  995 CONTINUE
C      IF((ITN-1)/100*100.EQ.(ITN-1)) WRITE (6,3002)
C      IF(ITN .EQ. 0) WRITE (6,3002)
C!!!!!SELECT INTERVAL AT WHICH OUTPUT OF RESIDUALS IS DESIRED
C      IF((ITN-1)/10*10.EQ.(ITN-1)) WRITE (6,3001) RMAX,RUMAX,RVMAX,
C     1EMAX,IR,KR
      RETURN
 3002 FORMAT(//,4X,'DRMAX',11X,'DUMAX',11X,'DVMAX',11X,'DEMAX',10X,
     1'IR',3X,'KR')
 3001 FORMAT(4(E14.8,2X),2I5)
      END
C
C*************************************************************************
C
      SUBROUTINE MATRX1(IMAX,KMAX)
      COMMON/TRID/DD(4,4,161,41),MM(4,4,161,41),EE(4,4,161,41),
     1GG(4,161,41)
      COMMON/SCRAT/A(4,4,161),HH(4,4,161,41),C(4,5,161)
      REAL MM
      REAL L11,L21,L31,L41,L22,L32,L42,L33,L43,L44
     2,L1I,L2I,L3I,L4I
C
C     THIS SUBROUTINE PERFORMS THE BLOCK TRIDIAGONAL MATRIX IVERSION FOR
C     AN ENTIRE  PLANE DURING THE XI- SWEEP
C
      DO 1 I1 = 1, 4
      DO 1 K = 2 , KMAX - 1
      AI = 1. / MM(1,1,1,K)
      GG(I1,1,K) = GG(I1,1,K) * AI
      HH(I1,1,1,K) = EE(I1,1,1,K) * AI
      HH(I1,2,1,K) = EE(I1,2,1,K) * AI
      HH(I1,3,1,K) = EE(I1,3,1,K) * AI
      HH(I1,4,1,K) = EE(I1,4,1,K) * AI
    1 CONTINUE
C
```

```
      DO 1000 I = 2 , IMAX - 2
      DO 5    I1= 1 , 4
      DO 2    K = 2 , KMAX - 1
      C(I1,1,K)    = GG(I1,I,K) - DD(I1,1,I,K) * GG(1,I-1,K)
     1                          - DD(I1,2,I,K) * GG(2,I-1,K)
     2                          - DD(I1,3,I,K) * GG(3,I-1,K)
     3                          - DD(I1,4,I,K) * GG(4,I-1,K)
    2 CONTINUE
      DO 5    I2 = 1 , 4
      DO 5    K  = 2 , KMAX - 1
      A(I1,I2,K)= MM(I1,I2,I,K) - DD(I1,1,I,K) * HH(1,I2,I-1,K)
     1                          - DD(I1,2,I,K) * HH(2,I2,I-1,K)
     2                          - DD(I1,3,I,K) * HH(3,I2,I-1,K)
     3                          - DD(I1,4,I,K) * HH(4,I2,I-1,K)
      C(I1,I2+1,K)= EE(I1,I2,I,K)
    5 CONTINUE
      DO 3 K = 2 , KMAX - 1
      L11 = A(1,1,K)
      L1I = 1. / L11
      U12 = A(1,2,K) * L1I
      U13 = A(1,3,K) * L1I
      U14 = A(1,4,K) * L1I
      L21 = A(2,1,K)
      L31 = A(3,1,K)
      L41 = A(4,1,K)
      L22 = A(2,2,K) - L21 * U12
      L2I = 1. / L22
      U23 = (A(2,3,K) - L21 * U13) * L2I
      U24 = (A(2,4,K) - L21 * U14) * L2I
      L32 = A(3,2,K) - L31 * U12
      L42 = A(4,2,K) - L41 * U12
      L33 = A(3,3,K) - L31 * U13 - L32 * U23
      L3I = 1. / L33
      U34 = (A(3,4,K) - L31 * U14 - L32 * U24) * L3I
      L43 = A(4,3,K) - L41 * U13 - L42 * U23
      L44 = A(4,4,K) - L41 * U14 - L42 * U24 - L43 * U34
      L4I = 1. / L44
      C(1,1,K) = C(1,1,K) * L1I
      C(2,1,K) = (C(2,1,K) - L21 * C(1,1,K)) * L2I
      C(3,1,K) = (C(3,1,K) - L31 * C(1,1,K)
     1                     - L32 * C(2,1,K)) * L3I
      C(4,1,K) = (C(4,1,K) - L41 * C(1,1,K) - L42 * C(2,1,K)
     1                                      - L43 * C(3,1,K)) * L4I
      C(1,2,K) = C(1,2,K) * L1I
      C(2,2,K) = (C(2,2,K) - L21 * C(1,2,K)) * L2I
      C(3,2,K) = (C(3,2,K) - L31 * C(1,2,K)
     1                     - L32 * C(2,2,K)) * L3I
      C(4,2,K) = (C(4,2,K) - L41 * C(1,2,K) - L42 * C(2,2,K)
     1                                      - L43 * C(3,2,K)) * L4I
      C(1,3,K) = C(1,3,K) * L1I
      C(2,3,K) = (C(2,3,K) - L21 * C(1,3,K)) * L2I
      C(3,3,K) = (C(3,3,K) - L31 * C(1,3,K)
     1                     - L32 * C(2,3,K)) * L3I
      C(4,3,K) = (C(4,3,K) - L41 * C(1,3,K) - L42 * C(2,3,K)
     1                                      - L43 * C(3,3,K)) * L4I
      C(1,4,K) = C(1,4,K) * L1I
      C(2,4,K) = (C(2,4,K) - L21 * C(1,4,K)) * L2I
      C(3,4,K) = (C(3,4,K) - L31 * C(1,4,K)
     1                     - L32 * C(2,4,K)) * L3I
      C(4,4,K) = (C(4,4,K) - L41 * C(1,4,K) - L42 * C(2,4,K)
     1                                      - L43 * C(3,4,K)) * L4I
      C(1,5,K) =  C(1,5,K) * L1I
      C(2,5,K) = (C(2,5,K) - L21 * C(1,5,K)) * L2I
      C(3,5,K) = (C(3,5,K) - L31 * C(1,5,K)
     1                     - L32 * C(2,5,K)) * L3I
      C(4,5,K) = (C(4,5,K) - L41 * C(1,5,K) - L42 * C(2,5,K)
     1                                      - L43 * C(3,5,K)) * L4I
```

```
      C(3,1,K) = C(3,1,K) - U34 * C(4,1,K)
      C(2,1,K) = C(2,1,K) - U24 * C(4,1,K)
     1                    - U23 * C(3,1,K)
      C(1,1,K) = C(1,1,K) - U14 * C(4,1,K)
     1                    - U13 * C(3,1,K) - U12 * C(2,1,K)
      C(3,2,K) = C(3,2,K) - U34 * C(4,2,K)
      C(2,2,K) = C(2,2,K) - U24 * C(4,2,K)
     1                    - U23 * C(3,2,K)
      C(1,2,K) = C(1,2,K) - U14 * C(4,2,K)
     1                    - U13 * C(3,2,K) - U12 * C(2,2,K)
      C(3,3,K) = C(3,3,K) - U34 * C(4,3,K)
      C(2,3,K) = C(2,3,K) - U24 * C(4,3,K)
     1                    - U23 * C(3,3,K)
      C(1,3,K) = C(1,3,K) - U14 * C(4,3,K)
     1                    - U13 * C(3,3,K) - U12 * C(2,3,K)
      C(3,4,K) = C(3,4,K) - U34 * C(4,4,K)
      C(2,4,K) = C(2,4,K) - U24 * C(4,4,K)
     1                    - U23 * C(3,4,K)
      C(1,4,K) = C(1,4,K) - U14 * C(4,4,K)
     1                    - U13 * C(3,4,K) - U12 * C(2,4,K)
      C(3,5,K) = C(3,5,K) - U34 * C(4,5,K)
      C(2,5,K) = C(2,5,K) - U24 * C(4,5,K)
     1                    - U23 * C(3,5,K)
      C(1,5,K) = C(1,5,K) - U14 * C(4,5,K)
     1                    - U13 * C(3,5,K) - U12 * C(2,5,K)
    3 CONTINUE
C
      DO 6 I1       = 1 , 4
      DO 9 K        = 2 , KMAX - 1
    9 GG(I1,I,K)    = C(I1,1,K)
      DO 6 I2       = 1 , 4
      DO 6 K        = 2 , KMAX - 1
      HH(I1,I2,I,K) = C(I1,I2+1,K)
    6 CONTINUE
 1000 CONTINUE
C
C     BACKWARD SUBSTITUTION
C
      DO 7 I     = IMAX - 3, 1 , - 1
      DO 7 I1    = 1 , 4
      DO 7 K     = 2 , KMAX - 1
      GG(I1,I,K) = GG(I1,I,K) - HH(I1,1,I,K) * GG(1,I+1,K)
     1                        - HH(I1,2,I,K) * GG(2,I+1,K)
     2                        - HH(I1,3,I,K) * GG(3,I+1,K)
     3                        - HH(I1,4,I,K) * GG(4,I+1,K)
    7 CONTINUE
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE MATRX2(IMAX,KMAX)
      COMMON/TRID/DD(4,4,161,41),MM(4,4,161,41),EE(4,4,161,41),
     1GG(4,161,41)
      COMMON/SCRAT/A(4,4,161),HH(4,4,161,41),C(4,5,161)
      REAL MM
      REAL L11,L21,L31,L41,L22,L32,L42,L33,L43,L44
     2,L1I,L2I,L3I,L4I
C
C     THIS SUBROUTINE PERFORMS THE BLOCK TRIDIAGONAL MATRIX IVERSION FOR
C     AN ENTIRE J=CONSTANT PLANE DURING THE ZETA- SWEEP
C
      DO 1 I1       = 1, 4
      DO 1 I        = 2 , IMAX - 1
      AI            = 1. / MM(1,1,I,1)
      GG(I1,I,1)    = GG(I1,I,1) * AI
      HH(I1,1,I,1)  = EE(I1,1,I,1) * AI
```

```
      HH(I1,2,I,1) = EE(I1,2,I,1) * AI
      HH(I1,3,I,1) = EE(I1,3,I,1) * AI
      HH(I1,4,I,1) = EE(I1,4,I,1) * AI
    1 CONTINUE
C
      DO 1000 K = 2 , KMAX - 2
      DO 5    I1= 1 , 4
      DO 2    I = 2 , IMAX - 1
      C(I1,1,I) = GG(I1,I,K) - DD(I1,1,I,K) * GG(1,I,K-1)
     1                       - DD(I1,2,I,K) * GG(2,I,K-1)
     2                       - DD(I1,3,I,K) * GG(3,I,K-1)
     3                       - DD(I1,4,I,K) * GG(4,I,K-1)
    2 CONTINUE
      DO 5    I2 = 1 , 4
      DO 5    I  = 2 , IMAX - 1
      A(I1,I2,I)= MM(I1,I2,I,K) - DD(I1,1,I,K) * HH(1,I2,I,K-1)
     1                          - DD(I1,2,I,K) * HH(2,I2,I,K-1)
     2                          - DD(I1,3,I,K) * HH(3,I2,I,K-1)
     3                          - DD(I1,4,I,K) * HH(4,I2,I,K-1)
      C(I1,I2+1,I)= EE(I1,I2,I,K)
    5 CONTINUE
      DO 3 I = 2 , IMAX - 1
      L11 = A(1,1,I)
      L1I = 1. / L11
      U12 = A(1,2,I) * L1I
      U13 = A(1,3,I) * L1I
      U14 = A(1,4,I) * L1I
      L21 = A(2,1,I)
      L31 = A(3,1,I)
      L41 = A(4,1,I)
      L22 = A(2,2,I) - L21 * U12
      L2I = 1. / L22
      U23 = (A(2,3,I) - L21 * U13) * L2I
      U24 = (A(2,4,I) - L21 * U14) * L2I
      L32 = A(3,2,I) - L31 * U12
      L42 = A(4,2,I) - L41 * U12
      L33 = A(3,3,I) - L31 * U13 - L32 * U23
      L3I = 1. / L33
      U34 = (A(3,4,I) - L31 * U14 - L32 * U24) * L3I
      L43 = A(4,3,I) - L41 * U13 - L42 * U23
      L44 = A(4,4,I) - L41 * U14 - L42 * U24 - L43 * U34
      L4I = 1. / L44
      C(1,1,I) = C(1,1,I) * L1I
      C(2,1,I) = (C(2,1,I) - L21 * C(1,1,I)) * L2I
      C(3,1,I) = (C(3,1,I) - L31 * C(1,1,I)
     1                     - L32 * C(2,1,I)) * L3I
      C(4,1,I) = (C(4,1,I) - L41 * C(1,1,I) - L42 * C(2,1,I)
     1                                      - L43 * C(3,1,I)) * L4I
      C(1,2,I) = C(1,2,I) * L1I
      C(2,2,I) = (C(2,2,I) - L21 * C(1,2,I)) * L2I
      C(3,2,I) = (C(3,2,I) - L31 * C(1,2,I)
     1                     - L32 * C(2,2,I)) * L3I
      C(4,2,I) = (C(4,2,I) - L41 * C(1,2,I) - L42 * C(2,2,I)
     1                                      - L43 * C(3,2,I)) * L4I
      C(1,3,I) = C(1,3,I) * L1I
      C(2,3,I) = (C(2,3,I) - L21 * C(1,3,I)) * L2I
      C(3,3,I) = (C(3,3,I) - L31 * C(1,3,I)
     1                     - L32 * C(2,3,I)) * L3I
      C(4,3,I) = (C(4,3,I) - L41 * C(1,3,I) - L42 * C(2,3,I)
     1                                      - L43 * C(3,3,I)) * L4I
      C(1,4,I) = C(1,4,I) * L1I
      C(2,4,I) = (C(2,4,I) - L21 * C(1,4,I)) * L2I
      C(3,4,I) = (C(3,4,I) - L31 * C(1,4,I)
     1                     - L32 * C(2,4,I)) * L3I
      C(4,4,I) = (C(4,4,I) - L41 * C(1,4,I) - L42 * C(2,4,I)
     1                                      - L43 * C(3,4,I)) * L4I
      C(1,5,I) = C(1,5,I) * L1I
```

```
      C(2,5,I) = (C(2,5,I) - L21 * C(1,5,I)) * L2I
      C(3,5,I) = (C(3,5,I) - L31 * C(1,5,I)
     1                      - L32 * C(2,5,I)) * L3I
      C(4,5,I) = (C(4,5,I) - L41 * C(1,5,I) - L42 * C(2,5,I)
     1                                  - L43 * C(3,5,I)) * L4I
      C(3,1,I) = C(3,1,I) - U34 * C(4,1,I)
      C(2,1,I) = C(2,1,I) - U24 * C(4,1,I)
     1                    - U23 * C(3,1,I)
      C(1,1,I) = C(1,1,I) - U14 * C(4,1,I)
     1                    - U13 * C(3,1,I) - U12 * C(2,1,I)
      C(3,2,I) = C(3,2,I) - U34 * C(4,2,I)
      C(2,2,I) = C(2,2,I) - U24 * C(4,2,I)
     1                    - U23 * C(3,2,I)
      C(1,2,I) = C(1,2,I) - U14 * C(4,2,I)
     1                    - U13 * C(3,2,I) - U12 * C(2,2,I)
      C(3,3,I) = C(3,3,I) - U34 * C(4,3,I)
      C(2,3,I) = C(2,3,I) - U24 * C(4,3,I)
     1                    - U23 * C(3,3,I)
      C(1,3,I) = C(1,3,I) - U14 * C(4,3,I)
     1                    - U13 * C(3,3,I) - U12 * C(2,3,I)
      C(3,4,I) = C(3,4,I) - U34 * C(4,4,I)
      C(2,4,I) = C(2,4,I) - U24 * C(4,4,I)
     1                    - U23 * C(3,4,I)
      C(1,4,I) = C(1,4,I) - U14 * C(4,4,I)
     1                    - U13 * C(3,4,I) - U12 * C(2,4,I)
      C(3,5,I) = C(3,5,I) - U34 * C(4,5,I)
      C(2,5,I) = C(2,5,I) - U24 * C(4,5,I)
     1                    - U23 * C(3,5,I)
      C(1,5,I) = C(1,5,I) - U14 * C(4,5,I)
     1                    - U13 * C(3,5,I) - U12 * C(2,5,I)
    3 CONTINUE
C
      DO 6 I1       = 1 , 4
      DO 9 I        = 2 , IMAX - 1
    9 GG(I1,I,K)    = C(I1,1,I)
      DO 6 I2       = 1 , 4
      DO 6 I        = 2 , IMAX - 1
      HH(I1,I2,I,K) = C(I1,I2+1,I)
    6 CONTINUE
 1000 CONTINUE
C
C     BACKWARD SUBSTITUTION
C
      DO 7 K      = KMAX - 3, 1 , - 1
      DO 7 I1     = 1 , 4
      DO 7 I      = 2 , IMAX - 1
      GG(I1,I,K) = GG(I1,I,K) - HH(I1,1,I,K) * GG(1,I,K+1)
     1                        - HH(I1,2,I,K) * GG(2,I,K+1)
     2                        - HH(I1,3,I,K) * GG(3,I,K+1)
     3                        - HH(I1,4,I,K) * GG(4,I,K+1)
    7 CONTINUE
      RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE METRIC
      COMMON/FIX/OMEGA
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/GRID/YACOB(161,41)
      COMMON/MTR'X/XIX(161,41),XIZ(161,41),ZETAX(161,41),
     +ZETAZ(161,41),
     1XIT(161,41),ZETAT(161,41)
C
C***  SUBROUTINE METRIC COMPUTES THE METRICS IN BOTH DIRECTIONS AND
C     THE UNSTEADY COEFFICIENTS ETAT, ETC.
```

```
C
      DO 1000 K = 1 , KMAX
      DO 1000 I = 1 , IMAX
      XTAU = OMEGA * Z(I,K)
      YTAU = OMEGA * (-X(I,K))
C***  PRESENT SET UP IS FOR FLOW PAST AN  AIRFOIL.
C
C!!!!!CENTRAL DIFFERENCES AT INTERIOR POINTS, TWO-POINT ONE-SIDED
C!!!!!DIFFERENCES DOWNSTREAM, THREE-POINT AT OTHER OUTER BOUNDARIES
      IF(I.EQ.1.OR.I.EQ.IMAX) GO TO 10
      XXI = .5 * (X(I+1,K)-X(I-1,K))
      ZXI = .5 * (Z(I+1,K)-Z(I-1,K))
      GO TO 15
   10 IF(I.EQ.IMAX) GO TO 16
      XXI = 1.0 * (X(2,K) - X(1,K))
      ZXI = 1.0 * (Z(2,K) - Z(1,K))
      GO TO 15
   16 XXI = 1.0 * (X(IMAX,K) - X(IMAX-1,K))
      ZXI = 1.0 * (Z(IMAX,K) - Z(IMAX-1,K))
   15 CONTINUE
      IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 17
      XZET = .5 *(X(I,K+1)-X(I,K-1))
      ZZET = .5 *(Z(I,K+1)-Z(I,K-1))
      GO TO 20
  17  IF(K.EQ.KMAX) GO TO 18
      XZET = 2. * X(I,2)-1.5 * X(I,1) - .5 * X(I,3)
      ZZET = 2. * Z(I,2) - 1.5 * Z(I,1) - .5 * Z(I,3)
      GO TO 20
   18 XZET = 1.5 * X(I,KMAX)-2.* X(I,KMAX-1)+.5*X(I,KMAX-2)
      ZZET = 1.5 * Z(I,KMAX)-2.* Z(I,KMAX-1)+.5*Z(I,KMAX-2)
   20 CONTINUE
C!!!!!COMPUTE JACOBIAN
      YACOBI = XXI * ZZET - XZET * ZXI
      YACOB(I,K) = 1. / YACOBI
      XIX(I,K) = ZZET * YACOB(I,K)
      XIZ(I,K) = -XZET * YACOB(I,K)
      XTAU = OMEGA * Z(I,K)
      YTAU = - OMEGA * X(I,K)
      XIT(I,K) = - XIX(I,K) * XTAU - XIZ(I,K) * YTAU
      ZETAX(I,K) = -ZXI * YACOB(I,K)
      ZETAZ(I,K) = XXI * YACOB(I,K)
      ZETAT(I,K) = - ZETAX(I,K) * XTAU - ZETAZ(I,K) * YTAU
 1000 CONTINUE
      RETURN
      END
C
C**************************************************************************
C
      SUBROUTINE DISSIP
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      DIMENSION P(161),EPS(161),DIS1(161,4),DIS2(161,4)
C
C     THIS SUBROUTINE ADDS THE FOURTH ORDER DISSIPATION TERMS TO THE
C     RIGHT HAND SIDE
C
      IM1 = IMAX - 1
      KM1 = KMAX - 1
      IM2 = IMAX - 2
      KM2 = KMAX - 2
C
      DO 10 K=2 , KM1
C     COMPUTE SWTICHING FUNCTION BASED ON SECOND DERIVATIVE OF PRESSURE
      DO 1 I = 1 , IMAX
```

```
    1 P(I) = .4 * (Q4(I,K)-(Q2(I,K)**2+Q3(I,K)**2)/(2.*Q1(I,K)))
      DO 2 I =1 , IMAX
      IP2 = I + 2
      IF(I.EQ.IM1) IP2 = IMAX
      IM2 = I - 2
      IF(I.EQ.2) IM2 = 1
      IP1 = I + 1
      IF(I.EQ.IMAX) IP1 = IMAX
      IM = I - 1
      IF(I.EQ.1) IM = 1
      IF(I.EQ.1) IM2 = 1
      IF(I.EQ.IMAX) IP2 = IMAX
      EPS(I) = ABS(P(IP1)-2.*P(I)+P(IM))/ABS(P(IP1)+2.*P(I)+P(IM))
C     VOL = 2.  /(YACOB(I,K)+YACOB(IP1,K))
      VOL = 1.
      DIS1(I,1) =  (Q1(IP1,K)-Q1(I,K))*VOL
      DIS1(I,2) =  (Q2(IP1,K)-Q2(I,K))*VOL
      DIS1(I,3) =  (Q3(IP1,K)-Q3(I,K))*VOL
      HP1 = Q4(IP1,K)+P(IP1)
      HP = Q4(I,K)+P(I)
      HM1 = Q4(IM,K) + P(IM)
      HP2 = Q4(IP2,K) + P(IP2)
      HPM = Q4(IM,K)+P(IM)
      DIS1(I,4) =  (HP1-HP)*VOL
      DIS2(I,1) =- (Q1(IP2,K)-3.*(Q1(IP1,K)-Q1(I,K))-Q1(IM,K))*VOL
      DIS2(I,2) =- (Q2(IP2,K)-3.*(Q2(IP1,K)-Q2(I,K))-Q2(IM,K))*VOL
      DIS2(I,3) =- (Q3(IP2,K)-3.*(Q3(IP1,K)-Q3(I,K))-Q3(IM,K))*VOL
      DIS2(I,4) =- (HP2-3.*(HP1-HP)-HPM)*VOL
    2 CONTINUE
      DO 15 I = 1 , IM1
      D2P = AMAX1(EPS(I),EPS(I+1))
      C22 = 60. * D2P
      C11 = AMAX1(0.0,(1.-C22))
      C22 = C22 * WW/YACOB(I,K) * DT
      C11 = C11 * WW/YACOB(I,K) * DT
C!!!!!SWITCH ON SECOND-ORDER AND SWITCH OFF FOURTH-ORDER DISSIPATION
C!!!!!IN VICINITY OF SHOCKS
      DIS1(I,1) = C11 * DIS2(I,1) + C22 * DIS1(I,1)
      DIS1(I,2) = C11 * DIS2(I,2) + C22 * DIS1(I,2)
      DIS1(I,3) = C11 * DIS2(I,3) + C22 * DIS1(I,3)
      DIS1(I,4) = C11 * DIS2(I,4) + C22 * DIS1(I,4)
   15 CONTINUE
      DO 16 I = 2 , IM1
      DQ1(I,K) = DIS1(I,1) - DIS1(I-1,1)
      DQ2(I,K) = DIS1(I,2) - DIS1(I-1,2)
      DQ3(I,K) = DIS1(I,3) - DIS1(I-1,3)
      DQ4(I,K) = DIS1(I,4) - DIS1(I-1,4)
   16 CONTINUE
   10 CONTINUE
C     K DIRECTION
C!!!!!FOURTH-ORDER DISSIPATION ONLY
      DO 30 I = 2 , IM1
      WT= 0.5 * DT * WW / YACOB(I,2)
      W3 = 0.5 * DT * WW / YACOB(I,KM1)
      DQ1(I,2) =WT* (Q1(I,1) - 2. * Q1(I,2) + Q1(I,3))
     1+DQ1(I,2)
      DQ2(I,2) =WT* (Q2(I,1) - 2. * Q2(I,2) + Q2(I,3))
     1+DQ2(I,2)
      DQ3(I,2) =WT* (Q3(I,1) - 2. * Q3(I,2) + Q3(I,3))
     1+DQ3(I,2)
      DQ4(I,2) =WT* (Q4(I,1) - 2. * Q4(I,2) + Q4(I,3))
     1+DQ4(I,2)
      WT= W3
      DQ1(I,KM1) =WT* (Q1(I,KM2) - 2. * Q1(I,KM1) + Q1(I,KMAX))
     1+DQ1(I,KM1)
      DQ2(I,KM1) =WT* (Q2(I,KM2) - 2. * Q2(I,KM1) + Q2(I,KMAX))
     1+DQ2(I,KM1)
```

```
      DQ3(I,KM1) =WT* (Q3(I,KM2) - 2. * Q3(I,KM1) + Q3(I,KMAX))
     1+DQ3(I,KM1)
      DQ4(I,KM1) =WT* (Q4(I,KM2) - 2. * Q4(I,KM1) + Q4(I,KMAX))
     1+DQ4(I,KM1)
      DO 35 K = 3 , KM2
      WT= - DT * WW / YACOB(I,K)
      DQ1(I,K) =WT* (Q1(I,K+2) - 4. * Q1(I,K+1) + 6. * Q1(
     1I,K) - 4. * Q1(I,K-1) + Q1(I,K-2))+DQ1(I,K)
      DQ2(I,K) =WT*(Q2(I,K+2) - 4. * Q2(I,K+1) + 6. * Q2(
     1I,K) - 4. * Q2(I,K-1) + Q2(I,K-2))+DQ2(I,K)
      DQ3(I,K) =WT*(Q3(I,K+2) - 4. * Q3(I,K+1) + 6. * Q3(
     1I,K) - 4. * Q3(I,K-1) + Q3(I,K-2))+DQ3(I,K)
      DQ4(I,K) =WT*(Q4(I,K+2) - 4. * Q4(I,K+1) + 6. * Q4(
     1I,K) - 4. * Q4(I,K-1) + Q4(I,K-2))+DQ4(I,K)
   35 CONTINUE
   30 CONTINUE
C
      RETURN
      END
C
C*************************************************************************
C
      SUBROUTINE WALLBC
      COMMON/SURF/PSUR(161)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      COMMON/MTRIX/XIX(161,41),XIZ(161,41),ZETAX(161,41),
     +ZETAZ(161,41),
     1XIT(161,41),ZETAT(161,41)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      DIMENSION C1(4)
      DIMENSION A(2,2),RHS(2)
C!!!!!APPLY BOUNDARY CONDITIONS ON THE CUT AND THE AIRFOIL SURFACE
      DO 9 I=ITEU,IMAX
      I1 = IMAX + 1 - I
      Q1(I,1) = .5 * (Q1(I,2)+Q1(I1,2))
      Q2(I,1) =.5*(Q2(I,2)+Q2(I1,2))
      Q3(I,1) = .5 * (Q3(I,2) + Q3(I1,2))
      Q4(I,1) = .5 * (Q4(I,2)+Q4(I1,2))
      Q1(I1,1)=Q1(I,1)
      Q2(I1,1)=Q2(I,1)
      Q3(I1,1)=Q3(I,1)
      Q4(I1,1)=Q4(I,1)
    9 CONTINUE
      DO 1 I=  ITEL , ITEU
      K = 3
      C1(1) = XIT(I,K)
      C1(2) = XIX(I,K)
      C1(3) = XIZ(I,K)
      UCON3 = (Q2(I,K)*C1(2)+Q3(I,K)*C1(3))
     1/Q1(I,K)
      K = 2
      C1(1) = XIT(I,K)
      C1(2) = XIX(I,K)
      C1(3) = XIZ(I,K)
      UCON2 = (Q2(I,K)*C1(2)+Q3(I,K)*C1(3))
     1/Q1(I,K)
      RHS(1) = 2. * UCON2 - UCON3 - XIT(I,1)
C     FOR VISCOUS FLOWS SET UCON TO ZERO ALSO
      IF(REYREF.GT.0.) RHS(1) = - XIT(I,1)
      A(1,1) = XIX(I,1)
      A(1,2) = XIZ(I,1)
      A(2,1) = ZETAX(I,1)
      A(2,2) = ZETAZ(I,1)
```

```
      RHS(2) = - ZETAT(I,1)
      TEMP1 = A(1,1)
      TEMP2 = A(1,2)
      TEMP3 = A(2,1)
      TEMP4 = A(2,2)
      DEN = 1. /(TEMP1 * TEMP4 - TEMP2 * TEMP3)
      A(1,1) = A(2,2) * DEN
      A(1,2) = - TEMP2 * DEN
      A(2,1) = - TEMP3 * DEN
      A(2,2) = TEMP1 * DEN
      Q1(I,1) = 2. * Q1(I,2) - Q1(I,3)
      Q2(I,1) = Q1(I,1)*(A(1,1)*RHS(1)+A(1,2)*RHS(2))
      Q3(I,1) = Q1(I,1) *(A(2,1)*RHS(1)+A(2,2)*RHS(2))
    1 CONTINUE
      DO 10 I=ITEL ,ITEU
      U2=Q2(I,2)/Q1(I,2)
      W2=Q3(I,2)/Q1(I,2)
      P2=(GAMMA-1.)*(Q4(I,2)-0.5*Q1(I,2)*(U2*U2+W2*W2))
      U3=Q2(I,3)/Q1(I,3)
      W3=Q3(I,3)/Q1(I,3)
      P3=(GAMMA-1.)*(Q4(I,3)-0.5*Q1(I,3)*(U3*U3+W3*W3))
      P1=(4.*P2-P3)/3.
      PSUR(I)=(GAMMA*P1-1.)/(.7*AMINF**2)
      U1=Q2(I,1)/Q1(I,1)
      W1=Q3(I,1)/Q1(I,1)
   10 Q4(I,1)=P1/(GAMMA-1.)+0.5*Q1(I,1)*(U1*U1+W1*W1)
      RETURN
      END
C
C*************************************************************************
C
      SUBROUTINE STRESS(ITN,DALFA)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/MUTUR/CMU(161,41)
      DIMENSION AA(161,41)
     1,RH1(161),RH2(161),RH3(161),RH4(161)
      COMMON/LOGIC/RSTRT,PITCH,PLUNGE
      LOGICAL RSTRT,PITCH,PLUNGE
      U(I,J) = Q2(I,J) / Q1(I,J)
      V(I,J) = Q3(I,J) / Q1(I,J)
C     THIS SUBROUTINE ADDS VISCOUS TERMS TO THE RIGHT HAND SIDE
      GOGM = GAMMA / (GAMMA - 1.)
      IF(ITN.GT.10.OR.(RSTRT)) CALL EDDY(DALFA)
C     COMPUTE U AND V
      KMAXM1 = KMAX - 1
      IMAXM1 = IMAX - 1
      PR = 1.
      DO 10 K = 1 , KMAX
      DO 10 I = 1 , IMAX
      E      = Q4(I,K) / Q1(I,K) - 0.5 * (U(I,K)**2+V(I,K)**2)
   10 AA(I,K) = GOGM * E
C
C     COMPUTE TXX,TXY AND VISCOUS DISSIPATION AT I - 1 / 2
C
      DO 30 K = 2 , KMAXM1
      DO 20 I = 2 , IMAX
      UXI = U(I,K) - U(I-1,K)
      VXI = V(I,K) - V(I-1,K)
      AXI = AA(I,K) - AA(I-1,K)
      UZET= .25*(U(I,K+1)-U(I,K-1)+U(I-1,K+1)-U(I-1,K-1))
      VZET= .25*(V(I,K+1)-V(I,K-1)+V(I-1,K+1)-V(I-1,K-1))
      AZET= .25*(AA(I,K+1)-AA(I,K-1)+AA(I-1,K+1)-AA(I-1,K-1))
      XXI = X(I,K) - X(I-1,K)
```

```
      ZXI = Z(I,K) - Z(I-1,K)
      XZET= .25 * (X(I,K+1)-X(I,K-1)+X(I-1,K+1)-X(I-1,K-1))
      ZZET= .25 * (Z(I,K+1)-Z(I,K-1)+Z(I-1,K+1)-Z(I-1,K-1))
      YAC = XXI * ZZET - ZXI * XZET
      YAC = 1. / YAC
      XIX = ZZET * YAC
      ZETAX= - ZXI * YAC
      XIZ = -XZET * YAC
      ZETAZ= XXI * YAC
      CNM = .5 * (CMU(I,K) + CMU(I-1,K))
      UX  = UXI * XIX + UZET * ZETAX
      VX  = VXI * XIX + VZET * ZETAX
      AX  = AXI * XIX + AZET * ZETAX
      UZ  = UXI * XIZ + UZET * ZETAZ
      VZ  = VXI * XIZ + VZET * ZETAZ
      AZ  = AXI * XIZ + AZET * ZETAZ
      TXX = -(-4. * UX + 2. * VZ) * CNM / 3.
      TXY = CNM * (UZ + VX)
      TYY = -CNM / 3. * (-4. * VZ + 2. * UX)
      R4  = ((U(I,K)+U(I-1,K))*TXX+(V(I,K)+V(I-1,K))*TXY)*0.5
     1      + CNM / PR/(GAMMA - 1.) * AX
      S4  = ((U(I,K)+U(I-1,K))*TXY+(V(I,K)+V(I-1,K))*TYY)*0.5
     1      + CNM / PR / (GAMMA - 1.) * AZ
C       DEBUG
C       TURN OFF ENRGY DISSIPATION AND DIFFUSION
      R4 = 0.
      S4 = 0.
      RH1(I) = 0.
      RH2(I) = (XIX * TXX + XIZ * TXY) / YAC
      RH3(I) = (XIX * TXY + XIZ * TYY) / YAC
   20 RH4(I) = (XIX * R4 + XIZ * S4) / YAC
      DO 30 I = 2 , IMAXM1
      DQ1(I,K) = DQ1(I,K) + RH1(I+1) - RH1(I)
      DQ2(I,K) = DQ2(I,K) + RH2(I+1) - RH2(I)
      DQ3(I,K) = DQ3(I,K) + RH3(I+1) - RH3(I)
      DQ4(I,K) = DQ4(I,K) + RH4(I+1) - RH4(I)
   30 CONTINUE
C     IN THE Z DIRECTION
      DO 70 I = 2 , IMAXM1
      DO 60 K = 2 , KMAX
      UXI = .25 * (U(I+1,K)-U(I-1,K)+U(I+1,K-1)-U(I-1,K-1))
      VXI = .25 * (V(I+1,K)-V(I-1,K)+V(I+1,K-1)-V(I-1,K-1))
      AXI = .25 * (AA(I+1,K)-AA(I-1,K)+AA(I+1,K-1)-AA(I-1,K-1))
      XXI = .25 * (X(I+1,K)-X(I-1,K)+X(I+1,K-1)-X(I-1,K-1))
      ZXI = .25 * (Z(I+1,K)-Z(I-1,K)+Z(I+1,K-1)-Z(I-1,K-1))
      UZET = U(I,K) - U(I,K-1)
      VZET = V(I,K) - V(I,K-1)
      AZET = AA(I,K) - AA(I,K-1)
      XZET = X(I,K) - X(I,K-1)
      ZZET = Z(I,K) - Z(I,K-1)
      YAC = XXI * ZZET - ZXI * XZET
      YAC = 1. / YAC
      XIX = ZZET * YAC
      ZETAX= - ZXI * YAC
      XIZ = -XZET * YAC
      ZETAZ= XXI * YAC
      CNM = .5 * (CMU(I,K) + CMU(I,K-1))
      UX  = UXI * XIX + UZET * ZETAX
      VX  = VXI * XIX + VZET * ZETAX
      AX  = AXI * XIX + AZET * ZETAX
      UZ  = UXI * XIZ + UZET * ZETAZ
      VZ  = VXI * XIZ + VZET * ZETAZ
      AZ  = AXI * XIZ + AZET * ZETAZ
      TXX = -(-4. * UX + 2. * VZ) * CNM / 3.
      TXY = CNM * (UZ + VX)
      TYY = -CNM / 3. * (-4. * VZ + 2. * UX)
      R4  = ((U(I,K)+U(I,K-1))*TXX+(V(I,K)+V(I,K-1))*TXY)*0.5
```

```
     1      + CNM / PR/(GAMMA - 1.) * AX
      S4  = ((U(I,K)+U(I,K-1))*TXY+(V(I,K)+V(I,K-1))*TYY)*0.5
     1      + CNM / PR / (GAMMA - 1.) * AZ
      R4 = 0.
      S4 = 0.
      RH1(K) = 0.
      RH2(K) = (ZETAX * TXX + ZETAZ * TXY) / YAC
      RH3(K) = (ZETAX * TXY + ZETAZ * TYY) / YAC
   60 RH4(K) = (ZETAX * R4 + ZETAZ * S4) / YAC
      DO 70 K = 2 , KMAXM1
      DQ1(I,K) = DQ1(I,K) + RH1(K+1) - RH1(K)
      DQ2(I,K) = DQ2(I,K) + RH2(K+1) - RH2(K)
      DQ3(I,K) = DQ3(I,K) + RH3(K+1) - RH3(K)
      DQ4(I,K) = DQ4(I,K) + RH4(K+1) - RH4(K)
   70 CONTINUE
C
      RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE LOAD(CPS,CL,CD,CM,ALFAS)
      COMMON/GRID1/X(161,41),Y(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      DIMENSION CPS(161)
C
C     THIS SUBROUTINE COMPUTES THE INVISCID CONTRIBUTIONS
C     TO LOADS ON THE AIRFOIL SURFACE
C
      IMAXM1 = IMAX - 1
      CL = 0.
      CD = 0.
      CM = 0.
      DO 400 I = ITEL , ITEU - 1
      XL = .5 * (X(I,1)+X(I+1,1))
      YL = .5 * (Y(I,1)+Y(I+1,1))
      DX = X(I+1,1) - X(I,1)
      DY = Y(I+1,1) - Y(I,1)
      CPA = CPS(I+1) * .5 + CPS(I) * .5
      DCL = CPA * (-DX)
      DCD = CPA * DY
      CL = CL + DCL
      CD = CD + DCD
  400 CM = CM + DCD * YL - DCL * XL
C
      DCL = CL * COS(ALFAS) - CD * SIN(ALFAS)
      CD = CL * SIN(ALFAS) + CD * COS(ALFAS)
      CL = DCL
      RETURN
      END
C
C***********************************************************************
C
      SUBROUTINE WRAP(II,JJ,XSING,YSING,XP,YP,S0,A0,Y0)
      DIMENSION S0(161,4),Y0(41,4),A0(161,4),XP(1),YP(1)
C
C     THIS SUBROUTINE UNWRAPS THE AIRFOIL AND STORES THE UNWRAPPED
C     SURFACE IN ARRAYS A0 AND S0. IT ALSO DETERMINES THE STRETCHING
C     IN THE ETA DIRECTION.
C
      IMID = (II + 1) / 2
      DY   = .8 / (JJ - 2)
      DO 1 J = 2 , JJ
      Y = FLOAT(J-2) * DY
    1 Y0(J,1) = 1.25 * Y / (1. - Y * Y)
      Y0(1 , 1) = - Y0(3,1)
      PI = 4. * ATAN ( 1.)
```

```
      ANGL = PI + PI
      U = XP(1) - XSING
      V = YP(1) - YSING
      U = 1.
      V = 0.
      IIM1 = II - 1
      DO 2 I = 1 , II
      X11 = XP(I) - XSING
      Y11 = YP(I) - YSING
      ANGL = ANGL + ATAN2((U*Y11-V*X11),(U*X11+V*Y11))
      R   = SQRT(X11**2 + Y11**2)
      U   = X11
      V   =Y11
      R = SQRT(R)
      A0(I,1) = R * COS(.5 * ANGL)
    2 S0(I,1) = R * SIN(.5 * ANGL)
C!!!!!IF OUTPUT OF UNWRAPPED COORDINATES IS DESIRED
C      WRITE (6,1000)
C      WRITE (6,2000) (I,A0(I,1),S0(I,1),I = 1 , II)
      RETURN
 1000 FORMAT(1X,'UNWRAPPED COORDINATES IN THE TRANSFORMED PLANE')
 2000 FORMAT(I5 , 2F20.8)
      END
C
C************** ***********************************************************
C
      SUBROUTINE TABINT(XP,YP,XSING,YSING,N)
      DIMENSION XP(161),YP(161),S0(161),A0(161)
C!!!!!SMOOTH THE AIRFOIL SURFACE BY FINDING ADDITIONAL POINTS
      U = XP(1) - XSING
      V = YP(1) - YSING
      U = 1.
      V = 0.
      ANGL = 8. * ATAN(1.)
      DO 1 I = 1,N
      X11 = XP(I) - XSING
      Y11 = YP(I) - YSING
      ANGL = ANGL + ATAN2((U*Y11-V*X11),(U*X11+V*Y11))
      R = SQRT(X11**2 + Y11 ** 2)
      U = X11
      V = Y11
      R = SQRT(R)
      A0(I) = R * COS(ANGL * .5)
    1 S0(I) = R * SIN(ANGL * .5)
      DX =(A0(N)-A0(1))/96.
      A0ST = A0(1)
      DO 3 I = 1 , 97
      XX = FLOAT(I-1) * DX + A0ST
      CALL TAINT(A0,S0,XX,YY,N,3,NER,MON)
      XP(I) = XX * XX - YY * YY + XSING
    3 YP(I) = 2. * XX * YY + YSING
      RETURN
      END
C
C**************************************************************************
C
      SUBROUTINE TAINT(XTAB,FTAB,X,FX,N,K,NER,MON)
      DIMENSION XTAB(1),FTAB(1),T(10),C(10)
C
C     NASA - AMES SUBROUTINE FOR POLYNOMIAL INTERPOLATION
C     OF A TABULATED FUNCTION
C
      IF(N-K) 1 , 1 , 2
    1 NER = 2
      RETURN
    2 IF(K-9) 3,3,1
    3 IF(MON) 4,4,5
```

```
    5 IF(MON-2) 6,7,4
    4 J = 0
      NM1 = N - 1
      DO 8 I = 1 , NM1
      IF(XTAB(I) - XTAB(I+1)) 9,11,10
   11 NER = 3
      RETURN
    9 J = J-1
      GO TO 8
   10 J = J+1
    8 CONTINUE
      MON = 1
      IF(J) 12 , 6 , 6
   12 MON = 2
    7 DO 13 I = 1 , N
      IF(X - XTAB(I)) 14,14,13
   14 J = I
      GO TO 18
   13 CONTINUE
      GO TO 15
    6 DO 16 I = 1 , N
      IF(X-XTAB(I)) 16,17,17
   17 J = I
      GO TO 18
   16 CONTINUE
   15 J = N
   18 J = J - (K+1) / 2
      IF(J) 19,19,20
   19 J = 1
   20 M = J + K
      IF(M - N) 21,21,22
   22 J = J - 1
      GO TO 20
   21 KP1 = K + 1
      JSAVE = J
   26 DO 23 L = 1, KP1
      C(L) = X - XTAB(J)
      T(L) = FTAB(J)
   23 J = J+1
      DO 24 J = 1,K
      I = J+1
   25 T(I) = (C(J)*T(I)-C(I)*T(J))/(C(J)-C(I))
      I = I+1
      IF(I-KP1) 25,25,24
   24 CONTINUE
      FX = T(KP1)
      NER = 1
      RETURN
      END
C
C**********************************************************************
C
      SUBROUTINE SING(N2,N,X,Z,XLE,YLE,TEA,TES,XSING,YSING,CHD)
C
C
C     THIS SUBROUTINE COMPUTES SINGULAR POINT LOCATIONS.
C
      DIMENSION X(2) , Z(2)
      NU = N2
      N1 = N2 + 1
      N3  = N2 - 1
      X1  = X(N1)
      Z1  = Z(N1)
      X2  = X(N2)
      Z2  = Z(N2)
      X3  = X(N3)
      Z3  = Z(N3)
      D1  = X2 ** 2 - X1 ** 2
```

```
      D2  = Z2 ** 2 - Z1 ** 2
      D3  = X2 - X1
      D4  = Z2 - Z1
      D5  = X3 ** 2 - X1 ** 2
      D6  = Z3 ** 2 - Z1 ** 2
      D7  = X3 - X1
      D8  = Z3 - Z1
      B   = (D7 * ( D1 + D2) - D3*(D5+D6))/(2.*(D7*D4-D3*D8))
      IF(ABS(D3).LT.ABS(D7)) GO TO 10
      A = (D1 + D2 - 2. * B * D4) / (2. * D3)
      GO TO 20
   10 A = (D5 + D6 - 2. * B * D8) / ( 2. * D7)
   20 CONTINUE
      R = SQRT((X2-A)** 2 + (Z2-B)**2)
      XLE = X(NU)
      YLE = Z(NU)
      CHD = X(1) - X(NU)
      A2 = (Z(2)-Z(1)) /(X(2) - X(1))
      A1 = (Z(N)-Z(N-1))/(X(N)-X(N-1))
      TES = .5 * (A1 + A2)
      TEA = A2 - A1
      TEA = TEA * 57.29578
      XSING = (A+XLE) /2.
      YSING = (B+YLE) / 2.
      RETURN
      END
C
C**********************************************************************
C
      SUBROUTINE AIRFOL(II,JJ,IT,IE)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/YSYM/ISYM
      DIMENSION S0(161,4),A0(161,4),Y0(41,4),XP(161),YP(161),
     1E(161),F(161),B0(49)
C
      DATA (B0(I),I=1,32)/1.,1.0414,1.0836,1.1270,1.1715,1.2175,1.2651,
     11.3145,1.3659,1.4199,1.4755,1.5349,1.5973,1.6636,1.7342,1.8099,
     21.8914,1.9799,2.0764,2.1829,2.3012,2.4341,2.5653,2.7597,2.9646,
     33.2106,3.5141,3.9019,4.4219,5.1687,6.3632,8.6809/
C
C!!!!!COMPUTE THE COMPUTATIONAL GRID POINTS BASED ON INPUT AIRFOIL SHAPE
      DO 8 I = 1 , 32
    8 A0(I,1) = B0(I)
      READ (5,1)
    1 FORMAT(1X)
      READ (5,2) FNU,FNL,ZSYM
    2 FORMAT(3F10.0)
      ISYM = 0
      IF(ZSYM.NE.0.) ISYM = 1
*       II = 157
*       JJ =  41
*       IT =  31
*       IE = 127
      IIP1 = II + 1
      IIM1 = II - 1
      IIJJ = II * JJ
      IIJJ2 = II * ( JJ-2)
      ILE  = (IT + IE ) / 2
      ISTP = 0
      NN   = 5
      NRF  = 0
      NOTAPE = 1
      PI   = 4. * ATAN(1.)
      NU = FNU
      NL = FNL
      N  = NU + NL - 1
      READ(5,1)
```

```
      READ (5,333) (XP(I),YP(I),I = NL , N)
  333 FORMAT(2F10.0)
 9994 FORMAT(F20.8)
      L = N + 1
      IF(ZSYM .GT. 0.) GO TO 9995
      L = NL + 1
      READ(5,1)
      READ (5,333) (XP(L-I),YP(L-I),I=1,NL)
      GO TO 9996
 9995 K1 = L
      DO 16 I = NL , N
      K = K1 - I
      XP(K) = XP(I)
      YP(K) = - YP(I)
   16 CONTINUE
 9996 SCALE = 1. /(XP(1)-XP(NL))
      XX    = XP(NL)
      YY    = YP(NL)
      DO 9997 I = 1 , N
      XP(I) = XP(I) * SCALE
 9997 YP(I) = YP(I) * SCALE
      CALL SING(NU,N,XP,YP,XLE,ZLE,TEA,TES,XSING,YSING,CHD)
      CALL TABINT(XP,YP,XSING,YSING,N)
      NBODY = IE + 1 - IT
      DO 6791 I = 1 , NBODY
      L = I - 1
      E(IT+L) = XP(I)
 6791 F(IT+L) = YP(I)
      IEP1 = IE + 1
      SLOPT = TES * .75
      DO 438 I = IEP1 , II
      I1 = I +1 - IE
      E(I) = A0(I1,1)
      DXI = 1. / 48.
      D   = 4. / 3. * (E(I) - .25)
      F(I) = F(IE) + SLOPT * ALOG(D) / D
      L = IIP1 - I
      E(L) = E(I)
  438 F(L) = F(IT) + SLOPT * ALOG(D)/D
C      WRITE (6,439)
  439 FORMAT(2X,3H  I,19X,1HX,19X,1HY)
C      WRITE (6,37) (I,E(I),F(I),I = 1 , II)
      CALL WRAP(II,JJ,XSING,YSING,E,F,S0,A0,Y0)
C!!!!!MAP GRID BACK TO PHYSICAL PLANE AND SHIFT TO QUARTER CHORD
      DO 10 J = 2 , JJ
      DO 10 I = 1 , II
      X(I,J-1) = A0(I,1)**2 - (S0(I,1)+Y0(J,1))**2
     1 - 0.25
   10 Z(I,J-1) = 2. * A0(I,1) * (S0(I,1)+Y0(J,1))
      JJ = JJ - 1
      RETURN
   37 FORMAT(I5,2F20.8)
      END
C
C*************************************************************************************
C
      SUBROUTINE CLUSTR(DS)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      DIMENSION S(41),XP(41),YP(41),R(41)
C
C      THIS SUBROUTINE CLUSTERS A GIVEN X,Z GRID SUCH THAT THE FIRST POINT IS AT
C      THE USER-SPECIFIED DISTANCE DNMIN
C!!!!!COMPUTE THE OLD STRETCHING
      DO 100 I = 1 , IMAX
      S(1) = 0.
      XP(1) = X(I,1)
```

```
      YP(1) = Z(I,1)
      DO 10 K = 2 , KMAX
      XP(K) = X(I,K)
      YP(K) = Z(I,K)
   10 S(K) = SQRT((XP(K)-XP(K-1))**2+(YP(K)-YP(K-1))**2)
     1+S(K-1)
      SUMDX = S(KMAX)
      CALL STRTCH(SUMDX,DS,F1,KMAX,FACTOR)
C      WRITE (6,200) I,FACTOR
      R(1) = 0.
      DR = DS
      DO 20 K = 2 , KMAX
      R(K) = R(K-1) + DR
      DR = DR * FACTOR
   20 CONTINUE
      RLAST = 1. / R(KMAX)
      DO 30 K = 2 , KMAX
      R1 = R(K) * RLAST * S(KMAX)
C!!!!!REDISTRIBUTE THE CONSTANT-ETA LINES
      CALL TAINT(S,XP,R1,XP1,KMAX,3,NER,MON)
      X(I,K) = XP1
      CALL TAINT(S,YP,R1,YP1,KMAX,3,NER,MON)
      Z(I,K) = YP1
   30 CONTINUE
  100 CONTINUE
C      WRITE (6,115)
      DO 110 I = 1 , IMAX
      DX = X(I,2) - X(I,1)
      DY = Z(I,2) - Z(I,1)
      DN = SQRT(DX*DX+DY*DY)
C      WRITE(6,120) I , DX , DY , DN
  110 CONTINUE
      RETURN
  115 FORMAT(5X,6HNORMAL,1X,8HDISTANCE,3H AT,4H THE,5H WALL,/
     1,5H    I,8X,2HDX,8X,2HDY,8X,2HDN,//)
  120 FORMAT(I5,3F10.5)
  200 FORMAT(I5,F10.5)
      END
C
C*****************************************************************************
C
      SUBROUTINE STRTCH(SUMDX,DX1,F1,N1,R)
C
C     THIS SUBROUTINE DETERMINES A GEOMETRIC
C     PROGRESSION OF GRID SPACING BETWEEN 1 AND N1 SUCH THAT
C     SUM8DX) EQUALS SUMDX. THE RATIO BETWEEN SUCCESSIVE
C     SPACINGS IS R.
      N = N1 - 1
      R = 1.5
      E1 = 1.E-04
      E2 = 1.E-04
      DO 10 L = 1, 50
      F= (R-1) * SUMDX - DX1*(R**N-1)
      FP = SUMDX - DX1 * FLOAT(N) * R ** (N-1)
      RITER = R - F/ FP
C     IF(1.E-02.LT.RITER.AND.RITER.LT.1.) RITER = 1.
C     IF(1..LT.RITER.AND.RITER.LT.100.) RITER=.01
      IF(ABS(R-RITER).LT. R*E1) GO TO 1
      R = RITER
   10 CONTINUE
      R = 1.0001
      DX1 = DZTOT/FLOAT(N1-1)
      RETURN
    1 R= RITER
      RETURN
      END
C
```

```
C*****************************************************************************
C
      SUBROUTINE EDDY(DALFA)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/MUTUR/CMU(161,41)
      COMMON/SKINCF/CF(161)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/GRID1/X(161,41),Z(161,41)
      DIMENSION TIN(41),TOUT(41),Y(41)
C
C       INITIALIZE VISCOSITY EVERYWHERE
      FACT1 = DT * AMINF / REYREF
      CMUMAX = 100. * FACT1 / DT
      DO 1 K = 1 , KMAX
      DO 1 I = 1 , IMAX
    1 CMU(I,K) = FACT1
C     THIS SUBROUTINE COMPUTES THE EDDY VISCOSITY BASED ON THE
C     BALDWIN-LOMAX TWO LAYER MODEL
C
      DO 100 I = 2 , IMAX - 1
      UDIF = 0.
      FMAX = 0.1E-06
      YMAX = .1E-06
      FYMAX = 0.
      Y(1) = 0.
      UWALL = 0.
      IF(I.GT.ITEU.OR.I.LE.ITEL)UWALL = SQRT(Q2(I,1)**2+Q3(I,1)**2)/
     1Q1(I,1)
C     COMPUTE TAU AT THE WALL
      UET = 1.*(Q2(I,2)/Q1(I,2) - Q2(I,1)/Q1(I,1))
      VET = 1.*(Q3(I,2)/Q1(I,2) - Q3(I,1)/Q1(I,1))
      XXI = X(I+1,1) - X(I-1,1)
      ZXI = Z(I+1,1) - Z(I-1,1)
      XET = 4. * X(I,2) - 3. * X(I,1) - X(I,3)
      ZET = 4. * Z(I,2) - 3. * Z(I,1) - Z(I,3)
      XXI = .5 * XXI
      ZXI = .5 * ZXI
      XET = .5 * XET
      ZET = .5 * ZET
      YAC = 1. / (XXI * ZET - ZXI * XET)
      OMEGA =  (UET * XXI - VET * ZXI ) * YAC
      TWALL = AMINF * OMEGA / REYREF
      CF(I) = 2. * TWALL / (AMINF**2)
      FACT = SQRT(Q1(I,1) * ABS(TWALL))*REYREF/(26.*AMINF)
      DO 10 K = 2 , KMAX-1
      UXI = (Q2(I+1,K)/Q1(I+1,K) - Q2(I-1,K)/Q1(I-1,K))
      VXI = (Q3(I+1,K)/Q1(I+1,K)-Q3(I-1,K)/Q1(I-1,K))
      UET = (Q2(I,K+1)/Q1(I,K+1)-Q2(I,K-1)/Q1(I,K-1))
      VET = (Q3(I,K+1)/Q1(I,K+1)-Q3(I,K-1)/Q1(I,K-1))
      XXI = X(I+1,K) - X(I-1,K)
      ZXI = Z(I+1,K) - Z(I-1,K)
      XET = X(I,K+1) - X(I,K-1)
      ZET = Z(I,K+1) - Z(I,K-1)
      YAC = 1. / (XXI * ZET - ZXI * XET)
      OMEGA = ABS(UET*XXI+VET*ZXI-UXI*XET-VXI*ZET) * YAC
      UDIF = SQRT(Q2(I,K)**2+Q3(I,K)**2)/Q1(I,K) - UWALL
      IF(ABS(UDIF).GT.UDIFMAX) UDIFMAX = ABS(UDIF)
      Y(K) = SQRT((X(I,K)-X(I,K-1))**2+(Z(I,K)-Z(I,K-1))**2)+Y(K-1)
      F = Y(K) * OMEGA
      IF((Y(K)*FACT).GT.20.) GO TO 31
      IF(I.GT.ITEL.AND.I.LT.ITEU) F = F * (1. - EXP(-Y(K)*FACT))
   31 CONTINUE
C
C     MODIFIED TURBULENCE MODEL APPLY FOR SPECIFIED RANGE OF ANGLES WHERE
C     FY IS USED T0 FIND THE SECOND PEAK VALUE OF F FUNCTION
C
```

```
      IF(ALFA.LT.ALFAI.AND.DALFA.GE.0.) THEN
        FY = F * Y(K)
        IF(FY.GT.FYMAX) THEN
          FYMAX = FY
          FMAX  = F
          YMAX  = Y(K)
        ENDIF
      ENDIF
      IF(ALFA.GE.ALFAI.OR.DALFA.LT.0.) THEN
        IF(F.GT.FMAX) THEN
          FMAX = F
          YMAX = Y(K)
        ENDIF
      ENDIF
      FCT = Y(K) * FACT
      IF(FCT.GT.20.) FCT = 20.
      FCT = ABS(FCT)
      EL = .4 * Y(K) * (1. - EXP(-FCT))
      TIN(K) = Q1(I,K) * EL * EL * OMEGA
      TIN(K) = ABS(TIN(K))
   10 CONTINUE
      KSWTCH = 0
      FWAKE = YMAX * FMAX
      F1 = .25 * YMAX * UDIF **2 / FMAX
      IF(F1.LT.FWAKE) FWAKE = F1
      DO 20 K = 2 , KMAX - 1
      FKLEB = 0.
      IF(ABS(Y(K)/YMAX).LT.1.E+04) THEN
      FKLEB = 1. / (1. + 5.5 * (0.3 * Y(K)/YMAX) ** 6)
      END IF
      TOUT(K) = .0168 * 1.6 * Q1(I,K) * FWAKE * FKLEB
      TOUT(K) = ABS(TOUT(K))
      IF(KSWTCH.NE.0) GO TO 20
      IF(TIN(K).GT.TOUT(K)) KSWTCH = K - 1
   20  CONTINUE
C!!!!!TOTAL VISCOSITY IS SUM OF LAMINAR AND INNER/OUTER LAYER AS APPROPRIATE
      DO 30 K = 2 , KMAX - 1
      IF(K.LE.KSWTCH) THEN
      CMU(I,K) = DT * (AMINF/REYREF + ABS(TIN(K)))
      ELSE
      CMU(I,K) = DT * (AMINF / REYREF + ABS(TOUT(K)))
      END IF
   30 CONTINUE
  100 CONTINUE
      RETURN
      END
C
C*****************************************************************************
C
      SUBROUTINE RESI(OMEGA)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      DIMENSION RHS(161,4)
      XTAU(I,K) = OMEGA * Z(I,K)
      YTAU(I,K) = - OMEGA * X(I,K)
C     THIS SUBROUTINE COMPUTES THE RESIDUAL ON THE RIGHT HAND
C     SIDE ARISING FROM THE EULER- PART OF THE GOVERNING EQUATIONS
C
C    FLUX TERMS WITHIN THE XI- DERIVATIVE
      DO 100 K = 2 , KMAX - 1
      DO 10 I = 1 , IMAX
      UCON = (Q2(I,K)/Q1(I,K) ) * (Z(I,K+1)-Z(I,K-1))
     1 - (Q3(I,K)/Q1(I,K)) * (X(I,K+1)-X(I,K-1))
      UCON = 0.25 * DT * UCON
```

```
      XIT = - XTAU(I,K) *(Z(I,K+1)-Z(I,K-1))
     1 + YTAU(I,K) * (X(I,K+1) - X(I,K-1))
      XIT = XIT * DT * 0.25
      UCON = UCON + XIT
      RHS(I,1) = UCON * Q1(I,K)
      R = 1. / Q1(I,K)
      P = (GAMMA-1.) * (Q4(I,K) - .5 * R *(Q2(I,K)**2+
     1                                      Q3(I,K)**2))
      RHS(I,2) = Q2(I,K) * UCON + P * DT * 0.25 * (Z(I,K+1) - Z(I,K-1))
      RHS(I,3) = Q3(I,K) * UCON - P * DT * 0.25 * (X(I,K+1)-X(I,K-1))
      RHS(I,4) = UCON * (Q4(I,K)+P) - XIT * P
   10 CONTINUE
      DO 11 I = 2 , IMAX - 1
      DQ1(I,K) = DQ1(I,K) - RHS(I+1,1) + RHS(I-1,1)
      DQ2(I,K) = DQ2(I,K) - RHS(I+1,2) + RHS(I-1,2)
      DQ3(I,K) = DQ3(I,K) - RHS(I+1,3) + RHS(I-1,3)
   11 DQ4(I,K) = DQ4(I,K) - RHS(I+1,4) + RHS(I-1,4)
  100 CONTINUE
C
C      FLUX TERMS WITHIN THE ETA- DERIVATIVE
C
       DO 200 I = 2 , IMAX - 1
      DO 20 K = 1 , KMAX
      VCON = (Q2(I,K)/Q1(I,K) ) * (Z(I-1,K)-Z(I+1,K))
     1      +(Q3(I,K)/Q1(I,K) ) * (X(I+1,K)-X(I-1,K))
      VCON = VCON * 0.25 * DT
      ETAT = -XTAU(I,K) * (Z(I-1,K)-Z(I+1,K)) - YTAU(I,K)*
     1                                        (X(I+1,K)-X(I-1,K))
      ETAT = ETAT * 0.25 * DT
      VCON = VCON + ETAT
      RHS(K,1) = VCON * Q1(I,K)
      P = (GAMMA-1.) * (Q4(I,K) - 0.5 *(Q2(I,K)**2+Q3(I,K)**2)/Q1(I,K))
      RHS(K,2) = VCON * Q2(I,K) +P * DT * .25 * (Z(I-1,K)-Z(I+1,K))
      RHS(K,3) = VCON * Q3(I,K) + P * DT * .25 * (X(I+1,K) - X(I-1,K))
      RHS(K,4) = VCON * (Q4(I,K)+P) - ETAT * P
   20 CONTINUE
      DO 21 K = 2 , KMAX - 1
      DQ1(I,K) = DQ1(I,K) - RHS(K+1,1) + RHS(K-1,1)
      DQ2(I,K) = DQ2(I,K) - RHS(K+1,2) + RHS(K-1,2)
      DQ3(I,K) = DQ3(I,K) - RHS(K+1,3) + RHS(K-1,3)
   21 DQ4(I,K) = DQ4(I,K) - RHS(K+1,4) + RHS(K-1,4)
  200 CONTINUE
      RETURN
      END
C
C****************************************************************************
C
      SUBROUTINE ROTGRID(X,Z,IMAX,KMAX,DALFA)
C     ROTATE GRID IN THE CLOCKWISE DIRECTION BY AN AMOUNT DALFA
      DIMENSION X(161,41),Z(161,41)
      CA = COS(DALFA)
      SA = - SIN(DALFA)
      DO 20 K = 1 , KMAX
      DO 20 I = 1 , IMAX
      X1 = X(I,K)
      Z1 = Z(I,K)
      X(I,K) = X1 * CA - Z1 * SA
   20 Z(I,K) = Z1 * CA + X1 * SA
      RETURN
      END
C
C**************************************************************************
C
      SUBROUTINE CPPLOT(I1,I2,FMACH,X,Y,CP)
C
C     THIS SUBROUTINE PLOTS CP AT EQUAL INTERVALS IN THE MAPPED PLANE
C
```

```
      COMMON/SKINCF/CF(161)
      DIMENSION KODE(4),LINE(90),X(161),Y(161),CP(161)
      DIMENSION CFX(3,49),CFY(3,49),CPX(3,49),CPY(3,49)
      DATA KODE/1H ,1H+,1HI,1H*/
*      WRITE ( 6 , 2)
*    2 FORMAT(50H0PLOT OF CP AT EQUAL INTERVALS IN THE MAPPED PLANE/
*     1       10H0     X/C   ,10H     CPU   ,10H     CFU   ,10H     CFL   )
      CP0 = (1. + .2 * FMACH **2) ** 3.5 - 1.
      CP0 = CP0 / ( .7 * FMACH **2)
      K0 = 30. * CP0 + 4.5
      IMIN = (I2-I1)/2 + I1
      ILOW = 2 * IMIN
      ICOUNT = 0
      CHD=X(I1) - X(IMIN)
      DO 12 I = 1 , 90
   12 LINE(I) = KODE(1)
      DO 34 I = IMIN   , I2
      K = 30. * (CP0 - CP(I)) + 4.5
      K1 = 30. * (CP0 -CP(ILOW-I)) + 4.5
      IF(K.LT.1) K = 1
      IF(K1.LT.1) K1 = 1
      IF(K.GT.90) K = 90
      IF(K1 .GT. 90) K1 = 90
      LINE(K0) = KODE(3)
      LINE(K) = KODE(2)
      LINE(K1) = KODE(4)
      XOC = (X(I) - X(IMIN)) / CHD
*      WRITE (6,610) XOC,CP(I),CF(I),CF(ILOW-I),LINE
      LINE(K1) = KODE(1)
   34 LINE(K) = KODE(1)
C***  GENERATE PLOT3D CP AND CF PLOTTING FILES
      DO 500 I=IMIN,I2
           XOC = (X(I) - X(IMIN))/CHD
           ICOUNT = ICOUNT + 1
           CPY(1,ICOUNT) = 0.000000
           CFY(1,ICOUNT) = 0.000000
           CPY(2,ICOUNT) = CP(ILOW - I)
           CFY(2,ICOUNT) = CF(ILOW-1)
           CPY(3,ICOUNT) = CP(I)
           CFY(3,ICOUNT) = CF(I)
           CPX(1,ICOUNT) = XOC
           CFX(1,ICOUNT) = XOC
           CPX(2,ICOUNT) = XOC
           CFX(2,ICOUNT) = XOC
           CPX(3,ICOUNT) = XOC
           CFX(3,ICOUNT) = XOC
500   CONTINUE
      IDM = 3
      JDM = I2 - IMIN + 1
      WRITE(50) IDM,JDM
      WRITE(50)((CPX(I,J), I=1,IDM),J=1,JDM),
     +          ((CPY(I,J), I=1,IDM),J=1,JDM)
      WRITE(60) IDM,JDM
      WRITE(60)((CFX(I,J), I=1,IDM),J=1,JDM),
     +          ((CFY(I,J), I=1,IDM),J=1,JDM)
      RETURN
  610 FORMAT(4F10.4,90A1)
      END
/EOF
```

```
TITLE:
NACA 0012 AIRFOIL
IMAX:   KMAX:   DT:      WW:      ALFA:    ALFA1:   ALFAI:   REDFRE: AMINF:
161     41      .005     5.       15.00    10.00    19.00    0.151    .283
ITEL:   ITEU:   REYREF:  DNMIN:   TSTART:  FORMAT:  RSTRT:   PITCH:  PLUNGE:
31      127     3.45     .00005   -1.0     3.0      TRUE     TRUE     FALSE
CSTP:   CPLT:   NSTP:    PSTP:
0       0       1000     50
FNU:        FNL:        FSYM:
33.         33.         1.
  X:          Y:
0.          0.
0.0050       .01153
 .0125       .01894
 .0250       .02615
 .0500       .03555
 .0750       .04200
 .1000       .04683
 .1500       .05345
 .2000       .05737
 .2500       .05941
 .2600       .05962
 .2700       .05978
 .2800       .05992
 .2900       .05999
 .3000       .06000
 .3100       .05999
 .3200       .05992
 .3300       .05980
 .3400       .05965
 .3500       .05947
 .4000       .05803
 .4500       .05581
 .5000       .05294
 .5500       .04952
 .6000       .04563
 .6500       .04137
 .7000       .03664
 .7500       .03161
 .8000       .02623
 .8500       .02055
 .9000       .01448
 .9500       .00807
1.0000       .00126
```

LIST OF REFERENCES

1. NACA Technical Memorandum 678, *Increase in the Maximum Lift of an Airplane Due to a Sudden Increase in its Effective Angle of Attack Resulting from a Gust*, by M. Kramer, 1932.

2. Ham, N. D., and Garelick, M. S., "Dynamic Stall Considerations in Helicopter Rotors," *Journal of the American Helicopter Society* , v. 13, no. 2, pp. 49-55, April 1968.

3. Herbst, W. B., "Dynamics of Air Combat," *Journal of Aircraft* v. 20, no. 7, pp. 594-598, July 1983.

4. Davis, S. S., and Chang, I., "The Critical Role of Computational Fluid Dynamics in Rotary Wing Aerodynamics," *Vertica*, v. 11, no. 1 2, pp. 43-63, 1987.

5. American Institute of Aeronautics and Astronautics Paper 85-0129, *Numerical Solution of Unsteady Viscous Flow Past Rotor Sections*, by N. L. Sankar and W. Tang, 14-17 January 1985.

6. Tang, W., *Numerical Solutions of Unsteady Flow Past Rotor Sections*, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, Georgia, August 1986.

7. Bradshaw, P., Cebeci, T. and Whitelaw, J., *Engineering Calculation Methods for Turbulent Flow*, Academic Press, 1981.

8. Kuethe, A. and Chow, C., *Foundations of Aerodynamics*, John Wiley and Sons, 1976.

9. Watson, V., et al., *Use of Computer Graphics for Visualization of Flow Fields* , paper presented at AIAA Aerospace Engineering Conference, Los Angeles, California, February 1987.

10. Foley, J. D. and Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Co., 1982.

11. Shapiro, A. H., *The Dynamics and Thermodynamics of Compressible Fluid Flow, Vol. I*, John Wiley and Sons, Inc., 1953.

12. Sterling Software Technical Publication, *Graphics Animation System Version 2 User's Guide*, by G. Bancroft and F. Merritt, June, 1987.

13. National Aeronautics and Space Administration Technical Note D-8382, *Analysis of the Development of Dynamic Stall Based on Oscillating Airfoil Experiments*, by L. W. Carr, K. W. McAlister and W. J. McCroskey, January 1977.

14. United Technologies Research Center Report, *Airfoil Dynamic Stall at Constant Pitch Rate and High Reynolds Number*, by P. F. Lorber and F. O. Carta, 1987.

15. Francis, M. S., and Keesee, J. E., "Airfoil Dynamic Stall Performance with Large Amplitude Motions," *AIAA Journal*, v. 23, no. 11, pp. 1653-1659, November 1985.

16. Harris, F. D. and Pruyn, R. R., "Blade Stall - Half Fact, Half Fiction," *Journal of the American Helicopter Society*, v. 13, no. 2, pp. 27-48, April 1968.

17. Valdes, J. F., *Dynamic Stall Calculations Using a Navier-Stokes Solver*, MSAE Thesis, Naval Postgraduate School, Monterey, California, December, 1986.

18. Cowles, L., *High Reynold's Number Low Mach Number Steady Flow Field Calculations Over a NACA 0012 Airfoil Using Navier-Stokes and Interactive Boundary Layer Theory*, MSAE Thesis, Naval Postgraduate School, Monterey, California, December, 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman Department of Aeronautics and Astronautics, Code 67 Naval Postgraduate School Monterey, CA 93943	5
4.	Curricular Officer Department of Aeronautics and Astronautics, Code 31 Naval Postgraduate School Monterey, CA 93943	1
5.	Chief Fluid Mechanics Laboratory Mail Stop 260-1 NASA Ames Research Center Moffett Field, CA 94035	3
6.	Lawrence W. Carr Fluid Mechanics Laboratory NASA Ames Research Center Moffett Field, CA 94035	2
7.	Satyanarayana Bodapati Fluid Mechanics Laboratory NASA Ames Research Center Moffett Field, CA 94035	2
8.	Mr. Thomas Momiyama Head, Aircraft Division NAVAIR Code 931 Washington, D. C. 20361	1
9.	Dr. Turner Cebeci Professor and Chairman Dept. of Aerospace Engineering, CSULB Long Beach, CA 90840	1

10. Ms. Lisa Cowles 1
NADC Code 60C
Street Rd.
Warminster, PA 18974-5000

11. Mr. Harry Berman 1
NASC Headquarters Code 931M
Washington, D. C. 20361